CMPUT 365: Control with Approximation

Rupam Mahmood

April 4, 2022

Episodic semi-gradient prediction of q_π

In this chapter we return to the control problem, now with parametric approximation of the action-value function $\hat{q}(s, a, \mathbf{w}) \approx q_*(s, a)$, where $\mathbf{w} \in \mathbb{R}^d$ is a finite-dimensional weight

approximate q_*

The general gradient-descent update for action-value prediction is

$$\mathbf{w}_{t+1} \doteq \mathbf{w}_t + \alpha \Big[U_t - \hat{q}(S_t, A_t, \mathbf{w}_t) \Big] \nabla \hat{q}(S_t, A_t, \mathbf{w}_t). \tag{10.1}$$

gradient of the approximator.

For example, the update for the one-step Sarsa method is

$$\mathbf{w}_{t+1} \doteq \mathbf{w}_t + \alpha \Big[R_{t+1} + \gamma \hat{q}(S_{t+1}, A_{t+1}, \mathbf{w}_t) - \hat{q}(S_t, A_t, \mathbf{w}_t) \Big] \nabla \hat{q}(S_t, A_t, \mathbf{w}_t). \tag{10.2}$$

one step return.

next action's estimate.

We call this method *episodic semi-gradient one-step Sarsa*. For a constant policy, this method converges in the same way that TD(0) does, with the same kind of error bound (9.14).

Episodic semi-gradient control

Episodic Semi-gradient Sarsa for Estimating $\hat{q} \approx q_*$ (on-policy)

Input: a differentiable action-value function parameterization $\hat{q} : \mathcal{S} \times \mathcal{A} \times \mathbb{R}^d \to \mathbb{R}$
Algorithm parameters: step size $\alpha > 0$, small $\varepsilon > 0$
Initialize value-function weights $\mathbf{w} \in \mathbb{R}^d$ arbitrarily (e.g., $\mathbf{w} = \mathbf{0}$)

Loop for each episode:
  $S, A \leftarrow$ initial state and action of episode (e.g., ε-greedy)
  Loop for each step of episode:
    Take action A, observe R, S'
    If S' is terminal:
      $\mathbf{w} \leftarrow \mathbf{w} + \alpha\big[R - \hat{q}(S, A, \mathbf{w})\big]\nabla\hat{q}(S, A, \mathbf{w})$
      Go to next episode
    Choose A' as a function of $\hat{q}(S', \cdot, \mathbf{w})$ (e.g., ε-greedy)
    $\mathbf{w} \leftarrow \mathbf{w} + \alpha\big[R + \gamma\hat{q}(S', A', \mathbf{w}) - \hat{q}(S, A, \mathbf{w})\big]\nabla\hat{q}(S, A, \mathbf{w})$
    $S \leftarrow S'$
    $A \leftarrow A'$

prediction: use $\hat{q}$ to select action instead of $\hat{q}$
policy is fixed (doesn't improve).

Potential question: What are the key differences between episodic Sarsa for prediction vs. control?

Describe it by performing minimal changes to this pseudocode to achieve Sarsa for prediction

Mountain Car

$$\hat{q}(s,a,\mathbf{w}) \doteq \mathbf{w}^\top \mathbf{x}(s,a) = \sum_{i=1}^{d} w_i \cdot x_i(s,a),$$

position and velocity.

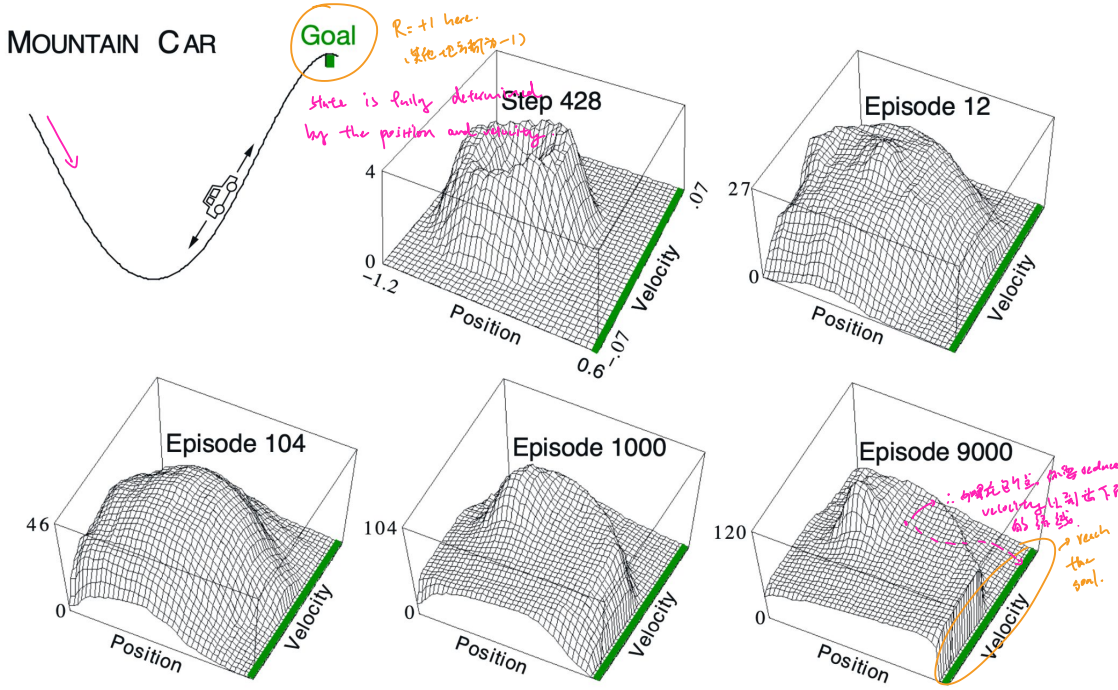


Figure 10.1: The Mountain Car task (upper left panel) and the cost-to-go function ($-\max_a \hat{q}(s,a,\mathbf{w})$) learned during one run.

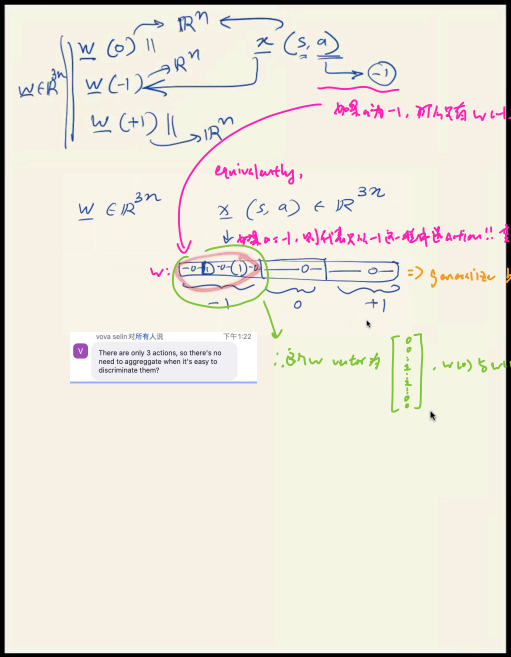
Reward is -1 every time step

Action is discretized

Termination is only when you reach the goal

For a specific position and velocity, if the car takes action -1, does the update at that step generalizes to other two actions for that same position and velocity?

In this example, there is generalization across states but not actions, which is still treated through lookup table!


$w \in \mathbb{R}^{3n}$
$w(0) \in \mathbb{R}^n$
$w(-1) \in \mathbb{R}^n$
$w(+1) \in \mathbb{R}^n$
$x(s, a)$
(-1)
如果a为-1，对应的 w(-1) 会被 updated! 其他w都不会变!!
equivalently,
$w \in \mathbb{R}^{3n}$
$x(s, a) \in \mathbb{R}^{3n}$
如果a=-1，则只有对应-1这一组会被activate!! 其他会被 activate 的概率=0!!
w:
-1
0
+1
=> generalize by coarse coding or tile coding.
vova selin 对所有人说 下午1:22
There are only 3 actions, so there's no need to aggregate when it's easy to discriminate them?
∴ 这时w vector为 [...], w(0) 与 w(+1) 都为 [...]

The average reward problem setting for continuing tasks

In continuing tasks, the interaction between agent and environment goes on and on forever without termination

In the average-reward setting, the quality of a policy π is defined as the average rate of reward, or simply *average reward*, while following that policy, which we denote as $r(\pi)$:

$$r(\pi) \doteq \lim_{h\to\infty} \frac{1}{h} \sum_{t=1}^{h} \mathbb{E}[R_t \mid S_0, A_{0:t-1} \sim \pi] \tag{10.6}$$

$$= \lim_{t\to\infty} \mathbb{E}[R_t \mid S_0, A_{0:t-1} \sim \pi], \tag{10.7}$$

$$= \sum_s \mu_\pi(s) \sum_a \pi(a|s) \sum_{s',r} p(s', r \mid s, a) r,$$

where the expectations are conditioned on the initial state, S_0, and on the subsequent actions, $A_0, A_1, \ldots, A_{t-1}$, being taken according to π. The second and third equations hold if the steady-state distribution, $\mu_\pi(s) \doteq \lim_{t\to\infty} \Pr\{S_t = s \mid A_{0:t-1} \sim \pi\}$, exists and is independent of S_0, in other words, if the MDP is *ergodic*. In an ergodic MDP, the

How is the quality of a policy determined in the discounted setting?

The average reward problem setting for continuing tasks (cont'd)

Note that the steady state distribution μ_π is the special distribution under which, if you select actions according to π, you remain in the same distribution. That is, for which

$$\sum_s \mu_\pi(s) \sum_a \pi(a|s) p(s'|s,a) = \mu_\pi(s'). \tag{10.8}$$

policy distribution π

take action from policy π

The outcome will be the same policy distribution!!

In the average-reward setting, returns are defined in terms of differences between rewards and the average reward:

$$G_t \doteq R_{t+1} - r(\pi) + R_{t+2} - r(\pi) + R_{t+3} - r(\pi) + \cdots. \tag{10.9}$$

This is known as the *differential* return, and the corresponding value functions are known as *differential* value functions. Differential value functions are defined in terms

The average reward problem setting for continuing tasks (cont'd)

known as *differential* value functions. Differential value functions are defined in terms of the new return just as conventional value functions were defined in terms of the discounted return; thus we will use the same notation, $v_\pi(s) \doteq \mathbb{E}_\pi[G_t|S_t = s]$ and $q_\pi(s,a) \doteq \mathbb{E}_\pi[G_t|S_t = s, A_t = a]$ (similarly for v_* and q_*), for differential value functions.

Differential value functions also have Bellman equations, just slightly different from those we have seen earlier. We simply remove all γs and replace all rewards by the difference between the reward and the true average reward:

$$v_\pi(s) = \sum_a \pi(a|s) \sum_{r,s'} p(s', r \,|\, s, a) \Big[r - r(\pi) + v_\pi(s') \Big],$$

$$q_\pi(s,a) = \sum_{r,s'} p(s', r \,|\, s, a) \Big[r - r(\pi) + \sum_{a'} \pi(a'|s') q_\pi(s', a') \Big],$$

$$v_*(s) = \max_a \sum_{r,s'} p(s', r \,|\, s, a) \Big[r - \max_\pi r(\pi) + v_*(s') \Big], \text{ and}$$

$$q_*(s,a) = \sum_{r,s'} p(s', r \,|\, s, a) \Big[r - \max_\pi r(\pi) + \max_{a'} q_*(s', a') \Big]$$

Differential semi-gradient Sarsa for ~~prediction~~ control

Differential semi-gradient Sarsa for estimating $\hat{q} \approx q_*$

Input: a differentiable action-value function parameterization $\hat{q} : \mathcal{S} \times \mathcal{A} \times \mathbb{R}^d \to \mathbb{R}$
Algorithm parameters: step sizes $\alpha, \beta > 0$, small $\varepsilon > 0$
Initialize value-function weights $\mathbf{w} \in \mathbb{R}^d$ arbitrarily (e.g., $\mathbf{w} = \mathbf{0}$)
Initialize average reward estimate $\bar{R} \in \mathbb{R}$ arbitrarily (e.g., $\bar{R} = 0$)

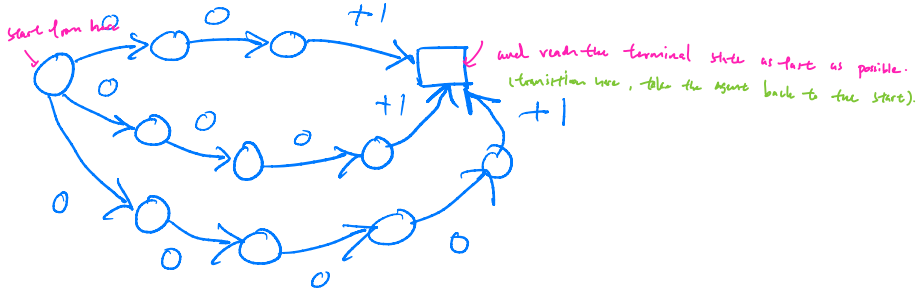
Initialize state S, and action A
Loop for each step:
  Take action A, observe R, S'
  Choose A' as a function of $\hat{q}(S', \cdot, \mathbf{w})$ (e.g., ε-greedy)
  $\delta \leftarrow R - \bar{R} + \hat{q}(S', A', \mathbf{w}) - \hat{q}(S, A, \mathbf{w})$
  $\bar{R} \leftarrow \bar{R} + \beta\delta$
  $\mathbf{w} \leftarrow \mathbf{w} + \alpha\delta\nabla\hat{q}(S, A, \mathbf{w})$
  $S \leftarrow S'$
  $A \leftarrow A'$

Control Approximation

Friday

[Use of discounted setting]

Discounted setting works well for many tasks:



In the above task, the shortest path can be connected with the optimal policy for <u>any</u> $\gamma < 1$.

Many tasks in our life is like that. Completing a degree, reaching a destination, etc. They may as well be formulated with -1 reward every timestep with $\gamma = 1$ because they are episodic.

[Average reward setting]

Similarly, in continuing task if we care about indefinitely long horizons there is an alternative to discounting setting, for which γ should be arbitrarily close to and yet less than 1 (see coursera).

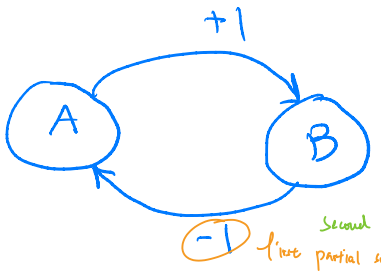
It is to maximize average reward:

$$r(\pi) = \lim_{h \to \infty} E_\pi \left[\frac{1}{h} \sum_{t=1}^{h} R_t \right].$$

Our algorithms are value based, which can still be retained by defining differential value functions:

$$v_\pi(s) \doteq E_\pi \left[\sum_{t=0}^{\infty} (R_{t+1} - r(\pi)) \mid S_0 = s \right]$$

$$= E_\pi \left[R_{t+1} - r(\pi) + v_\pi(S_{t+1}) \mid S_0 = s \right]$$



first partial sum.

Second partial sum: -1+1

Let's consider a Markov chain (doesn't have actions) where we don't have to worry about choosing a policy ($|A| = 1$).

($|A| = 1$: the set of actions only have one action.)

What is average reward?

Calculation of Average reward:

$$r(\pi) = \lim_{h \to \infty} \frac{1}{h} (1 - 1 + 1 - 1 + \cdots)$$

$$= \lim_{h \to \infty} c_h$$

$$c_1 = \frac{1}{1} = 1 \; ; \; c_2 = \frac{1-1}{2} = 0$$

$$c_3 = \frac{1-1+1}{3} = \frac{1}{3} \; , \; c_4 = 0$$

$$c_5 = \frac{1}{5} \; , \; c_7 = \frac{1}{7}$$

↓ even iteration: return = 0

odd iteration: return > 0.

what is $v_\pi(A)$?

$$v_\pi(A) = \underbrace{(1}_{\text{reward}} - \underbrace{0)}_{\text{discounted reward } (1/2)\cdot 0} + (-1-0) + (1-0) + \cdots$$
$$= 1-1+1-1+1-1+\cdots$$

This sum does not converge.

But one may come up with a reasonable meaning of this sum through estimation.

At every step, we look back at all the partial sums and form an average. In the limit, we will have an estimate of the full sum.

Say we are to estimate $C = a_1 + a_2 + a_3 + \cdots$

Partial sum up to time t: $c_t = \sum_{k=1}^{t} a_k$ (up to the t^{th} step, my sum is the sum of t different partial sums).

All partial sums at t: $c_1, c_2, \cdots, c_t$.

Then our estimate $\frac{c_1 + c_2 + \cdots + c_t}{t} = \frac{1}{t}\sum_{k=1}^{t} c_k$

$\lim_{t\to\infty} \frac{1}{t}\sum_{k=1}^{t} c_k$ then converges and called the **Cesàro sum** of series $a_1, a_2, a_3, \cdots$

The Cesàro sum for sequence $1,-1,+1,-1\cdots$

is $\lim_{t\to\infty} \frac{1}{t}\left(\underbrace{1 + (1-1) + (1-1+1) + (1-1+1-1) + \cdots}_{t \text{ partial sums}} \right)$

$= \lim_{t\to\infty} \frac{1}{t}\left(\underbrace{1+(1-1)}_{t/2 \text{ of them repeats}} + 1 + (1-1) + \cdots \right)$

$= \lim_{t\to\infty} \frac{1}{t} \times \frac{t}{2}\left(1 + 1-1\right) = \frac{1}{2}.$

(Also) $= \frac{1}{2}\left(\underbrace{(R_1 - r_\pi)}_{\text{first partial sum.}} + (R_1 - r_\pi) + \underbrace{(R_2 - r_\pi)}_{\text{second partial sum}} \right) = \frac{1}{2}\left(2(R_1 - r_\pi) + (R_2 - r_\pi) \right).$

this is 0 here.

(useful for such periodic sequence)

Note that unlike regular sum, Cesàro sum depends on the order of the sequence:

$$v_\pi(B) = -1+1-1+1+\cdots = \frac{1}{2}\left(2R_1^{-0} + R_2^{-0}\right)$$

$$= \frac{1}{2}(-2+1)$$

$$= -\frac{1}{2}.$$

[Abel sum]

Another way of getting the same estimation is through Abel sum:

$$\lim_{\gamma \to 1} \left(a_1 + \gamma a_2 + \gamma^2 a_3 + \gamma^3 a_4 + \cdots \right).$$

Then we can define a differential value function as:

$$v_\pi(s) \doteq \lim_{\gamma \to 1} E_\pi \left[\sum_{t=0}^{\infty} \gamma^t \left(R_{t+1} - r(\pi) \right) \middle| S_0 = s \right].$$

$$v_\pi(A) = \lim_{\gamma \to 1} \left(-\gamma + \gamma^2 - \gamma^3 + \cdots \right)$$

$$= \lim_{\gamma \to 1} \frac{1}{1+\gamma} = \frac{1}{2}.$$