The Value Iteration Algorithm

Alice Gao

Lecture 19

Readings: RN 17.2, 17.3. PM 9.5.2, 9.5.3.

Outline

Learning Goals

By the end of the lecture, you should be able to

- Trace the execution of and implement the value iteration algorithm for solving a Markov Decision Process.
- Trace the execution of and implement the policy iteration algorithm for solving a Markov Decision Process.

Learning Goals

Solving for $V^*(s)$ using Value Iteration

Policy Iteration

Revisiting the Learning goals

Solving for $V^*(s)$

V and Q are defined recursively in terms of each other.

$$V^*(s) = R(s) + \gamma \max_a Q^*(s,a) \tag{1}$$

$$Q^*(s,a) = \sum_{s'} P(s'|s,a) V^*(s'). \tag{2}$$

Combining equations 1 and 2, we get the Bellman equations:

$$V^*(s) = R(s) + \gamma \max_a \sum_{s'} P(s'|s,a) V^*(s'). \tag{3}$$

$V^*(s)$ are the unique solution to the Bellman equations.

Write down $V^*(s_{11})$

Write down the Bellman equation for $V^*(s_{11})$.

CQ: Solve the Bellman equations efficiently

CQ: Can we solve the system of Bellman equations efficiently?

(A) Yes

(B) No

(C) I don't know

The Bellman equation for $V^*(s_{11})$:

$$\begin{aligned} V^*(s_{11}) = -0.04 + \gamma \max[&0.8V^*(s_{12}) + 0.1V^*(s_{21}) + 0.1V^*(s_{11}), \\ &0.9V^*(s_{11}) + 0.1V^*(s_{12}), \\ &0.9V^*(s_{11}) + 0.1V^*(s_{21}), \\ &0.8V^*(s_{21}) + 0.1V^*(s_{12}) + 0.1V^*(s_{11})]. \end{aligned}$$

Solving for $V^*(s)$ iteratively

The Bellman equations:

$$V^*(s) = R(s) + \gamma \max_a \sum_{s'} P(s'|s,a) V^*(s').$$

Let $V_i(s)$ be the values for the i-th iteration.

1. Start with arbitrary initial values for $V_0(s)$.
2. At the i-th iteration, compute $V_{i+1}(s)$ as follows.

$$V_{i+1}(s) \leftarrow R(s) + \gamma \max_a \sum_{s'} P(s'|s,a) V_i(s')$$

3. Terminate when $\max_s |V_i(s) - V_{i+1}(s)|$ is small enough.

If we apply the Bellman update infinitely often,
the V_i's are guaranteed to converge to the optimal values.

Apply Value Iteration

Let's apply the value iteration algorithm.

Assume that

- the discount factor $\gamma = 1$.
- $R(s) = -0.04, \forall s \neq s_{24}, s \neq s_{34}$.

Start with $V_0(s) = 0, \forall s \neq s_{24}, s \neq s_{34}$.

CQ: Calculating $V_1(s_{23})$

CQ: What is $V_1(s_{23})$?

(A) $(-\infty, 0)$ (B) $[0, 0.25)$ (C) $[0.25, 0.5)$
(D) $[0.5, 0.75)$ (E) $[0.75, 1]$

$V_0(s)$:

	1	2	3	4
1	0	0	0	0
2	0	X	0	-1
3	0	0	0	+1

CQ: Calculating $V_1(s_{33})$

CQ: What is $V_1(s_{33})$?

(A) 0.26 (B) 0.36 (C) 0.46
(D) 0.56 (E) 0.76

$V_0(s)$:

	1	2	3	4
1	0	0	0	0
2	0	X	0	-1
3	0	0	0	+1

The Values of $V_1(s)$

$V_0(s)$:

	1	2	3	4
1	0	0	0	0
2	0	X	0	-1
3	0	0	0	+1

$V_1(s)$:

	1	2	3	4
1				
2		X		-1
3				+1

CQ: Calculating $V_2(s_{33})$

CQ: What is $V_2(s_{33})$?

(A) 0.822

(B) 0.832

(C) 0.842

(D) 0.852

(E) 0.862

CQ: Calculating $V_2(s_{23})$

CQ: What is $V_2(s_{23})$?

(A) 0.464

(B) 0.466

(C) 0.468

(D) 0.470

(E) 0.472

CQ: Calculating $V_2(s_{32})$

CQ: What is $V_2(s_{32})$?

(A) 0.16

(B) 0.36

(C) 0.56

(D) 0.76

(E) 0.96

The Values of $V_2(s)$

$V_1(s)$:

	1	2	3	4
1	-0.04	-0.04	-0.04	-0.04
2	-0.04	X	-0.04	-1
3	-0.04	-0.04	0.76	$+1$

$V_2(s)$:

	1	2	3	4
1				
2		X		-1
3				$+1$

Observations from Value Iteration

Each state accumulates negative rewards
until the algorithm finds a path to the $+1$ goal state.

How should we update $V^*(s)$ for all states s?

- synchronous: store and use $V_i(s)$ to calculate $V_{i+1}(s)$.
- asynchronous: stores $V_i(s)$ and update the values one at a time, in any order.

Policy Iteration

- Deriving the optimal policy does not require accurate estimates of the utility function ($V^*(s)$).
- Policy iteration alternates between two steps.
  - Policy evaluation: Given a policy π_i, calculate $V^{\pi_i}(s)$, which is the utility of each state if π_i were to be executed.
  - Policy improvement: Calculate a new policy π_{i+1} usingV^{π_i}.

Terminates when there is no change in the policy.

Policy Iteration

- Policy improvement:

$$\pi_{i+1}(s) = \arg\max_a \sum_{s'} P(s'|s,a) V^{\pi_i}(s').$$

- Policy evaluation:

$$V_i(s) = R(s) + \gamma \sum_{s'} P(s'|s,\pi_i(s)) V_i(s').$$

Policy Evaluation v.s. Bellman Equations

Policy evaluation:

$$V(s) = R(s) + \gamma \sum_{s'} P(s'|s, \pi(s)) V(s').$$

Bellman equations:

$$V(s) = R(s) + \gamma \max_{a} \sum_{s'} P(s'|s, a) V(s').$$

Write down both equations for $V(s_{11})$.
Assume that $\pi(s_{11}) =$ down.

Performing Policy Evaluation Exactly

Policy evaluation:

$$V(s) = R(s) + \gamma \sum_{s'} P(s'|s, \pi(s)) V(s').$$

Solve the system of linear equations exactly
using standard linear algebra techniques.

For n states, this will take $O(n^3)$ time.

Performing Policy Evaluation Iteratively

Policy evaluation:

$$V(s) = R(s) + \gamma \sum_{s'} P(s'|s, \pi(s)) V(s').$$

Solve the system of linear equations approximately
by performing a number of simplified value iteration steps.

Revisiting the Learning Goals

By the end of the lecture, you should be able to

- Trace the execution of and implement the value iteration algorithm for solving a Markov Decision Process.
- Trace the execution of and implement the policy iteration algorithm for solving a Markov Decision Process.