facebook
Artificial Intelligence Research

How to solve an MDP incrementally: Reinforcement Learning

Alessandro Lazaric
Facebook AI Research

Outline

Policy Evaluation

Fixed policy π

For $i = 1, \ldots, n$

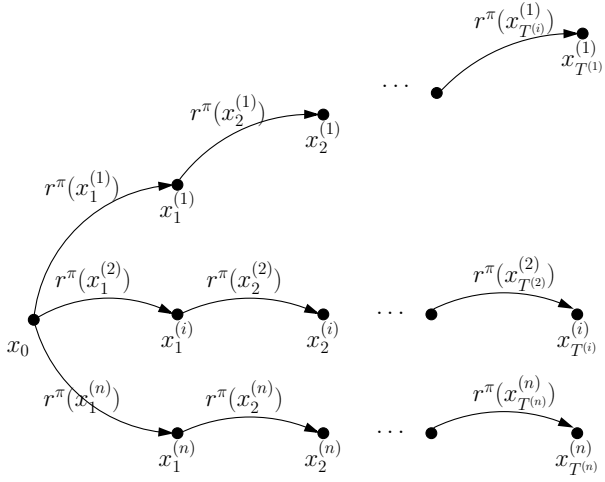
1. Set $t = 0$
2. Set initial state s_0
3. **While** ($s_{t,i}$ not terminal) *[execute one trajectory]*
   1. Take action $a_{t,i} = \pi(s_{t,i})$
   2. Observe **next state** $s_{t+1,i}$ and **reward** $r_{t,i} = r(s_{t,i}, a_{t,i}) \sim \nu(s_{t,i}, a_{t,i})$
   3. Set $t = t + 1$

   EndWhile

EndFor

Return: Estimate of the value function $\widehat{V}^{\pi}(\cdot)$

The RL Interaction Protocol



State Value Function

Fixed point of Bellman equation

$$V^{\pi}(s) = r^{\pi}(s) + \gamma \sum_{s'} p^{\pi}(s'|s) V^{\pi}(s')$$

Temporal Difference $TD(0)$

Input: π, T, s_0, V_0
$V = V_0$, $s = s_0$
for $t = 1, \ldots, T$ **do**
    Execute action $a_t = \pi(s_t)$
    Observe r_t and next state s_{t+1}
    Update

$$\widehat{V}(s_t) = \widehat{V}(s_t) + \alpha(s_t)\big(r_t + \gamma \widehat{V}(s_{t+1}) - \widehat{V}(s_t)\big)$$

end
return $\widehat{V}$

Temporal Difference $TD(0)$: Intuition

The **Temporal difference error** of estimate $\widehat{V}^\pi$ w.r.t. transition (s_t, r_t, s_{t+1})

$$\delta_t = r_t + \gamma \widehat{V}^\pi(s_{t+1}) - \widehat{V}^\pi(s_t)$$

Temporal Difference $TD(0)$: Intuition

The **Temporal difference error** of estimate $\widehat{V}^\pi$ w.r.t. transition (s_t, r_t, s_{t+1})

$$\delta_t = r_t + \gamma \widehat{V}^\pi(s_{t+1}) - \widehat{V}^\pi(s_t)$$

- Bellman error for function $\widehat{V}$ at state s

$$\mathcal{B}^\pi(\widehat{V}; s) = r^\pi(s) + \gamma \sum_{s'} p^\pi(s'|s)\widehat{V}(s') - \widehat{V}(s) \qquad [\mathcal{B}^\pi(V^\pi; s) = 0]$$

- Conditioned on s_t, δ_t is an *unbiased* estimate of $\mathcal{B}^\pi$

$$\mathbb{E}_{r_t, s_{t+1}}\left[\delta_t \middle| s_t\right] = r^\pi(s_t) + \gamma \mathbb{E}_{s_{t+1}|s_t}\left[\widehat{V}^\pi(s_{t+1})\right] - \widehat{V}^\pi(s_t) = \mathcal{B}^\pi(\widehat{V}^\pi, s_t)$$

- Expected dynamics of TD(0)

$$\overline{V}^\pi(s_t) = \overline{V}^\pi(s_t) + \alpha(s_t)\mathcal{B}^\pi(\overline{V}^\pi; s_t)$$

Temporal-Difference TD(0): Alternative View

- Mix between old and new estimate of $V^\pi(s_t)$

  old estimate $\widehat{V}^\pi(s_t)$ new estimate $r_t + \gamma \widehat{V}^\pi(s_{t+1})$

- **Weighted average**

$$\widehat{V}^\pi(s_t) = (1-\alpha)\widehat{V}^\pi(s_t) + \alpha\Big(r_t + \gamma \widehat{V}^\pi(s_{t+1})\Big)$$

Temporal-Difference TD(0): Properties

Theorem

Let TD(0) run with learning rate $\alpha(N_t(s_t))$ where $N_t(s_t)$ is the number of visits to the state s_t. If all states are visited infinitely often and the learning rate is set such that

$$\sum_{t=0}^{\infty} \alpha(t) = \infty \qquad \sum_{t=0}^{\infty} \alpha(t)^2 < \infty \qquad \textit{[Robbins Monro's condition]}$$

then for any state $s \in \mathcal{S}$

$$\widehat{V}^{\pi}(s) \xrightarrow{a.s.} V^{\pi}(s).$$

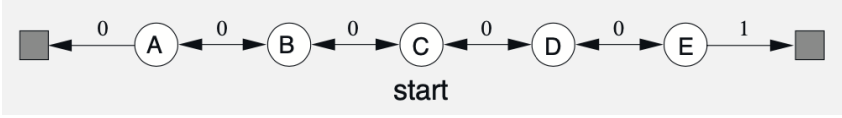
☞ standard choise is $\alpha_t(s) = \frac{1}{i^{\beta}}$ for $\beta \in (1/2, 1]$

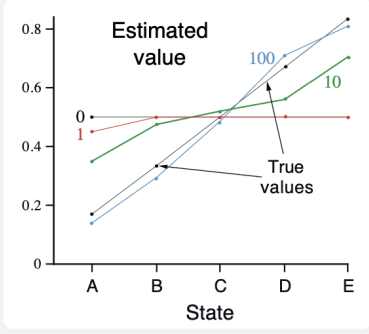
$$\widehat{V}_t(s) = \widehat{V}_{t-1}(s) + \frac{1}{N_t(s)^{\beta}}(r + \gamma \widehat{V}_{t-1}(s') - \widehat{V}_{t-1}(s))$$

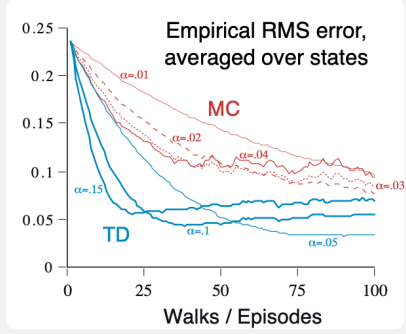
where $N_t(s)$ is the number of visits to state s before t

Example: Random Walk

(?)







Outline

Policy Learning

Learn optimal policy π^*

For $i = 1, \ldots, n$

1. Set $t = 0$
2. Set initial state s_0
3. **While** ($s_{t,i}$ not terminal) *[execute one trajectory]*
   1. **Take action** $a_{t,i}$
   2. Observe next state $s_{t+1,i}$ and reward $r_{t,i} = r(s_{t,i}, a_{t,i})$
   3. Set $t = t + 1$

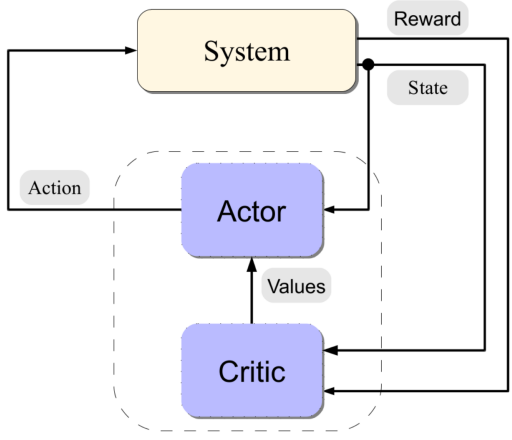
   EndWhile

EndFor

Return: Estimate of the optimal policy $\widehat{\pi}^*$

Actor-Critic Methods

- Critic: estimate the value function of the policy given by the actor
- Actor: change the policy to improve the value given by the critic



Policy Iteration

Let π_0 be an arbitrary stationary policy

while $k = 1, \ldots, K$ **do**

Policy Evaluation: given π_k compute Q^{π_k}

Policy Improvement: find π_{k+1} that is better than π_k

- e.g., compute the *greedy* policy

$$\pi_{k+1}(s) \in \arg\max_{a \in \mathcal{A}} Q^{\pi_k}(s, a)$$

return the last policy π_K

end

Policy Iteration is an instance of actor-critic methods:

- Actor: acts greedily (i.e., $\pi_{k+1} = \text{greedy}(Q^{\pi_k})$)
- Critic: compute Q given the policy π_k

Generalized Policy Iteration

More generally:

- Actor: *policy improvement* (even approximate)
- Critic: *policy evaluation*

(?) refers to actor/critic methods as **generalized policy iteration** (GPI)

many possible schemes!

SARSA

Actor: the goal is to move the policy toward the greedy policy w.r.t. to the critic value

Possible strategies

- greedy (does not explore enough!)
- *ϵ-greedy* policy $\pi_Q(a|s)$

$$
\begin{aligned}
a &= \mathcal{U}(\mathcal{A}) && \text{with probability } \epsilon \\
a &= \arg\max_{\overline{a}} Q(s, \overline{a}) && \text{with probability } 1-\epsilon
\end{aligned}
$$

- *soft-max (random) exploratory* policy with temperature τ

$$\pi_Q(a|s) = \frac{\exp(Q(s,a)/\tau)}{\sum_{a'} \exp(Q(s,a')/\tau)}$$

The higher $Q(s,a)$, the more probability to take action a in state s

SARSA

Critic: uses TD to update the estimated Q-function

- Compute the temporal difference on the trajectory $\langle s_t, a_t, r_t, s_{t+1}, a_{t+1} \rangle$ (with actions chosen according to $\pi_Q(a|s)$)

$$\delta_t = r_t + \gamma \widehat{Q}(s_{t+1}, a_{t+1}) - \widehat{Q}(s_t, a_t)$$

SARSA

Critic: uses TD to update the estimated Q-function

- Compute the temporal difference on the trajectory $\langle s_t, a_t, r_t, s_{t+1}, a_{t+1} \rangle$ (with actions chosen according to $\pi_Q(a|s)$)

$$\delta_t = r_t + \gamma \widehat{Q}(s_{t+1}, a_{t+1}) - \widehat{Q}(s_t, a_t)$$

- Update the estimate of Q as

$$\widehat{Q}(s_t, a_t) = \widehat{Q}(s_t, a_t) + \alpha(s_t, a_t)\delta_t$$

State Action Reward State Action (SARSA) update

SARSA

Input: T, s_0, Q_0
$Q = Q_0$, $s = s_0$, $a = \mathcal{U}(\mathcal{A})$
for $t = 1, \dots, T$ **do**
  Execute action $a_t \sim \pi_{t,Q}(\cdot|s_t)$
  Observe r_t and next state s_{t+1}
  $a_{t+1} \sim \pi_{t,Q}(\cdot|s_{t+1})$
  Update

$$\widehat{Q}(s_t, a_t) = \widehat{Q}(s_t, a_t) + \alpha(s_t)\big(r_t + \gamma \widehat{Q}(s_{t+1}, a_{t+1}) - \widehat{Q}(s_t, a_t)\big)$$

end
return $\widehat{Q}$, π_Q

Examples

- $\pi_{t,Q}(a|s) = \dfrac{\exp(Q(s,a)/\tau_t}{\sum_b \exp(Q(s,b)/\tau_t)}$
- $\pi_{t,Q}(a|s) = (1-\epsilon_t)\mathbb{1}\left(a_t = \arg\max_b Q(s,b)\right) + \epsilon_t/A$

SARSA: Properties (informal)

- The TD updates make $\widehat{Q}$ converge to Q^{π}
- The update of π_Q allows to improve the policy
- A decreasing temperature allows to become more and more greedy
  $\Rightarrow$ If $\tau \to 0$ with a proper rate, then $\widehat{Q} \to Q^*$ and $\pi_Q \to \pi^*$
- Similarly for ϵ-greedy

SARSA: Limitations

The actions a_t need to be selected according to the current Q

$\Rightarrow$ **On-policy learning**

Outline

The Optimal Bellman Equation

Proposition

The optimal value function Q^* (i.e., $Q^* = \max_\pi Q^\pi$) is the solution to the optimal Bellman equation:

$$Q^*(s,a) = r(s,a) + \gamma \sum_{s'} p(s'|s,a) \max_{a' \in A} Q^*(s',a').$$

Q-Learning

Input: T, s_0, Q_0
$Q = Q_0$, $s = s_0$
for $t = 1, \ldots, T$ **do**
    Execute action $a_t \sim \pi_t(s_t)$
    Observe r_t and next state s_{t+1}
    Update

$$\widehat{Q}(s_t, a_t) = \widehat{Q}(s_t, a_t) + \alpha(s_t, a_t)(r_t + \gamma \max_{a'} \widehat{Q}(s_{t+1}, a') - \widehat{Q}(s_t, a_t))$$

end
return $\widehat{Q}$, greedy($\widehat{Q}$)

The policy π_t is only required to cover all actions with non-zero probability

Q-Learning: Properties

Proposition

If the learning rate satisfies the Robbins-Monro conditions in all states $s, a \in S \times A$

$$\sum_{i=0}^{\infty} \alpha_i(s,a) = \infty, \qquad \sum_{i=0}^{\infty} \alpha_i^2(s,a) < \infty,$$

and all state-action pairs are tried infinitely often, then for all $s, a \in S \times A$

$$\widehat{Q}(s,a) \xrightarrow{a.s.} Q^*(s,a)$$

Remark: this is another example of R-M algorithm
Remark: "infinitely often" requires a steady exploration policy

Q-Learning vs. SARSA

Update rule

- *Q-Learning*

$$\widehat{Q}(s_t, a_t) = \widehat{Q}(s_t, a_t) + \alpha(r_t + \gamma \max_{a'} \widehat{Q}(s_{t+1}, a') - \widehat{Q}(s_t, a_t))$$

- *SARSA*

$$\widehat{Q}(s_t, a_t) = \widehat{Q}(s_t, a_t) + \alpha(r_t + \gamma \widehat{Q}(s_{t+1}, a_{t+1}) - \widehat{Q}(s_t, a_t))$$

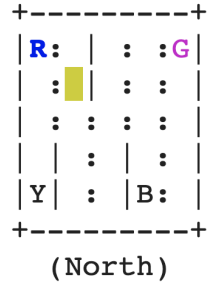
Both aim at learning the target policy $\pi^\star(s) = \arg\max_a Q^\star(s, a)$

- Q-learning can use any behavioral policy (*off-policy learning*)
- for SARSA the behavioral policy should be close to the estimated target policy (*on-policy learning*)

The Taxi Problem

Hands-on session [link] – [20min]

- Plot learning progress during training and evaluate performance at testing over multiple episodes
- How do learning rate and exploration parameters influence the performance?



Thank you!

facebook
Artificial Intelligence Research