Markov Decision Processes and Q-learning

Keith Ross

New York University

September 27, 2016

Outline for Today

- Bellman equation for MDPs
- Value Iteration
- Policy Improvement
- Q-learning
- Convergence proof for Q-learning

Useful Resources

- MDP book by Puterman
- DP books by Bertsekas
- Sutton and Barto (2nd Edition)
- Video lectures by David Silver
- John Tsitsiklis, Asynchronous stochastic approximation and Q-learning. Machine Learning, 1994

Discounted Criterion and Bellman Equation

- Recall the expected discounted reward

$$V_\pi(s) = E_\pi\left[\sum_{n=0}^{\infty} \beta^n r(S_n, A_n) \mid S_1 = s\right]$$

$$V^*(s) = \max_\pi V_\pi(s)$$

- Bellman Equations:

$$V_\pi(s) = r\left(s, \pi(s)\right) + \beta \sum_{s'} P_{ss'}\left(\pi(s)\right) V_\pi(s')$$

$$V^*(s) = \max_{a\in\mathcal{A}}\{r(s,a) + \beta \sum_{s'} P_{ss'}(a) V(s')\}$$

- The policy π that achieves the maxima is optimal. But how do we obtain $V^*(s)$?

Value Iteration Algorithm

- Initialize $V(s), s \in \mathcal{S}$ arbitrarily.
- $V(s) \leftarrow \max_{a \in \mathcal{A}} \{r(s,a) + \beta \sum_{s'} P_{ss'}(a) V(s')\}$
- Repeat until convergence

Synchronous and Asynchronous Updates:

- **Synchronous updates**: First compute the new values for $V(s)$ for every state s, then overwrite all the old values with the new ones.
- **Asynchronous updates**: Loop over the states, updating values one at a time.
- Both synchronous and asynchronous algorithms will converge to the optimal $V^*(s)$.

Outline of Proof for Value Iteration

Consider the metric space $(\mathcal{V}, d)$ where $\mathcal{V}$ contains all vectors

$$\mathbf{V} = (V(s), s \in \mathcal{S})$$

and d is the L^∞ norm defined as

$$d(\mathbf{V}, \mathbf{U}) = \max_{s \in \mathcal{S}} \big| V(s) - U(s) \big|$$

Define a mapping $\mathbf{T} : \mathcal{V} \to \mathcal{V}$:

$$\mathbf{T}(\mathbf{V})(s) = \max_{a} \{ r(s, a) + \beta \sum_{s'} P_{ss'}(a) V(s') \}$$

Can show that $\mathbf{T}$ is a contraction mapping, that is

$$d\left(\mathbf{T}(\mathbf{V}), \mathbf{T}(\mathbf{U})\right) \leq \beta d(\mathbf{V}, \mathbf{U})$$

for all $\mathbf{V}$ and $\mathbf{U}$. From the contraction mapping theorem, $\mathbf{T}(\mathbf{V}) = \mathbf{V}$ has a unique solution $\mathbf{V}^*$ and $\mathbf{T}^{(n)}(\mathbf{V}) \to \mathbf{V}^*$ for all initial $\mathbf{V}$, that is, value iteration converges to the solution of the Bellman equation.

Why is T a contraction?

$$
\begin{aligned}
d\left(\mathbf{T}(\mathbf{V}), \mathbf{T}(\mathbf{U})\right) &= \max_s \left| T(V)(s) - T(U)(s) \right| \\
&= \max_s \mid \max_a \{ r(s,a) + \beta \sum_{s'} P_{ss'}(a) V(s') \} \\
&\quad - \max_a \{ r(s,a) + \beta \sum_{s'} P_{ss'}(a) U(s') \} \mid \\
&\le \max_s \mid (r(s,a_s) + \beta \sum_{s'} P_{ss'}(a_s) V(s')) \\
&\quad - (r(s,a_s) + \beta \sum_{s'} P_{ss'}(a_s) U(s')) \mid \\
&= \beta \max_s \sum_{s'} P_{ss'}(a_s) \left| V(s') - U(s') \right| \\
&\le \beta \max_{s'} \left| V(s') - U(s') \right|
\end{aligned}
$$

Policy Improvement Algorithm

1. Initialize π arbitrarily.
2. Repeat until convergence {
   1. Solve $V(s) = r\left(s, \pi(s)\right) + \sum_{s'} P_{ss'}\left(\pi(s)\right) V(s')$
   2. $\pi(s) \leftarrow \arg\max_a \{r(s,a) + \sum_{s'} P_{ss'}(a) V(s')\}$

After a finite number of iterations, π will converge to optimal π^*.

Action-Value Function

Definition

The *Action-Value Function* $Q_\pi(s, a)$ is the long-run reward of starting in state s, taking action a, then following policy π.

$$Q_\pi(s, a) = E_\pi \left[\sum_{n=0}^{\infty} \beta^n r(S_n, A_n) \mid S_0 = s, A_0 = a \right]$$

The *Optimal Action-Value Function* is

$$Q^*(s, a) = \max_\pi Q_\pi(s, a)$$

Hence the optimal policy is

$$\pi^*(s) = \arg\max_a Q^*(s, a)$$

Reinforcement learning: Learn $Q^*(s, a)$ without knowing the environment $P_{ss'}(a)$.

Value Iteration for Action-Value Function

- Bellman equation for the action-value function:

$$Q^*(s,a) = r(s,a) + \beta \sum_{s'} P_{ss'}(a) \max_{a'} Q^*(s',a')$$

- Value iteration:

$$Q(s,a) \leftarrow r(s,a) + \beta \sum_{s'} P_{ss'}(a) \max_{a'} Q(s',a')$$

- Moving average version:

$$\begin{aligned} Q(s,a) \leftarrow & (1-\alpha) Q(s,a) + \alpha \left[r(s,a) + \beta \sum_{s'} P_{ss'}(a) \max_{a'} Q(s',a') \right] \\ = & Q(s,a) + \alpha \left[r(s,a) + \beta \sum_{s'} P_{ss'}(a) \max_{a'} Q(s',a') - Q(s,a) \right] \end{aligned}$$

Q-learning

- Recall value-iteration for action-value function:

$$Q(s,a) \leftarrow (1-\alpha)Q(s,a) + \alpha \left[r(s,a) + \beta \sum_{s'} P_{ss'}(a) \max_{a'} Q(s',a') \right]$$

$$= Q(s,a) + \alpha \left[r(s,a) + \beta \sum_{s'} P_{ss'}(a) \max_{a'} Q(s',a') - Q(s,a) \right]$$

- In RL we don't know $P_{ss'}(a)$ so we instead sample from it.

- Specifically, let $Q(s, a)$, $s \in \mathcal{S}, a \in \mathcal{A}$ be our current estimate of $Q^*(s, a)$.
- Suppose we are currently in state S, choose action $A = \arg\max_a Q(S, a)$ (plus some exploration TBD). Enter new state S' according to "environment".
- Update $Q(s, a)$ at (S, A):

$$
\begin{aligned}
Q(S, A) \leftarrow & (1 - \alpha) Q(S, A) + \alpha \left[R + \beta \max_{a'} Q(S', a') \right] \\
= & Q(S, A) + \alpha \left[R + \beta \max_{a'} Q(S', a') - Q(S, A) \right]
\end{aligned}
$$

Exploration and Off-Policy

ϵ-Greedy Policy

- With probability ϵ choose randomly from $\mathcal{A}$ (equally likely).
- With probability $1 - \epsilon$, choose "greedy":

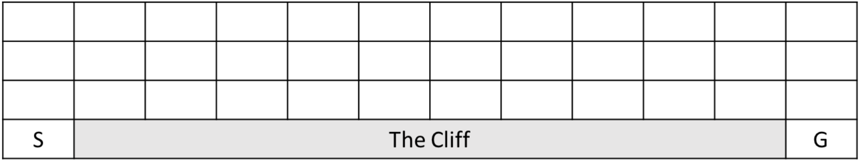
$$A = \arg\max_a Q(s, a)$$

Off-Policy:

- We can choose any rule for generating $(s_1, a_1, s_2, a_2, \dots)$.
- Only require s' to be generated according to $P_{ss'}(a)$.
- And all $s \in \mathcal{S}, a \in \mathcal{A}$ visited infinitely often.
- If step-size α is chosen appropriately, $Q(s, a)$ coverges to $Q^*(s, a)$ w.p.1

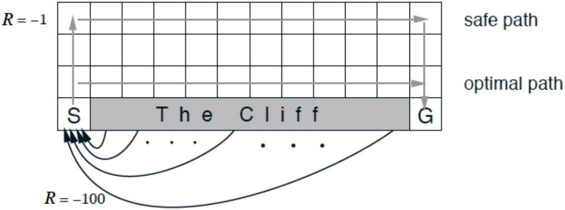
Example: Cliff Walking

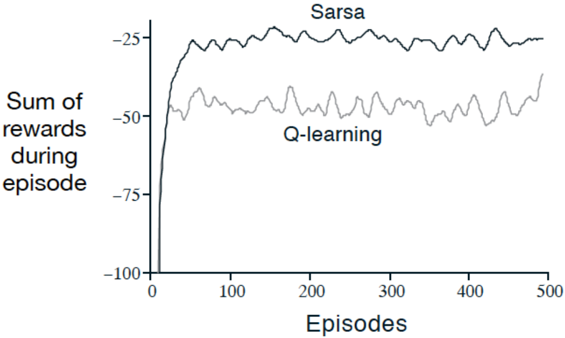
- Consider the gridworld shown in the figure below.
- The agent can move up, down, left, right.
- Reward is -1 on all transitions except those into the region marked "The Cliff".
- Stepping into this region incurs a reward of -100 and sends the agent instantly back to the start.



Source: Sutton, Barto: Reinforcement Learning - An Introduction, 2nd Edition Draft

Example: Cliff Walking



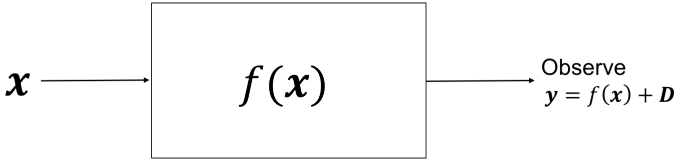


Example: Cliff Walking

- The graph in the previous slides shows the performance of Q-learning with ϵ-greedy action selection where $\epsilon = 0.1$.
- After an initial transient, Q-learning learns values for the optimal policy, that which travels right along the edge of the cliff.
- Unfortunately, this results in its occasionally falling off the cliff because of the ϵ-greedy action selection.
- After training, can use greedy policy ($\epsilon = 0$), so that you never fall off cliff.

Convergence Proof: Stochastic Approximations

Let $f : \mathbb{R}^d \to \mathbb{R}^d$ be a contraction. Suppose for given $\mathbf{x}$, we can only observe noisy values of $f(\mathbf{x})$.



$f(\mathbf{x})$ has unique fixed point: $f(\mathbf{x}^*) = \mathbf{x}^*$. How can we find it?

Robbins-Monro (1951)

Stochastic approximations algorithm:

$$\mathbf{x}_{n+1} = \mathbf{x}_n + \alpha_n[f(\mathbf{x}_n) + \mathbf{D}_n - \mathbf{x}_n]$$

- If $\sum_n \alpha_n = \infty$ and $\sum_n \alpha_n^2 < \infty$,
- and $\mathbf{D}_n$ is zero mean conditioned on the past (and another technical assumption),
- then $\mathbf{x}_n \to \mathbf{x}^*$ where $f(x^*) = x^*$.

What Does Stochastic Approximation Have to Do with Q-Learning?

- Let

$$f(\mathbf{Q})(s,a) \triangleq r(s,a) + \beta \sum_{s'} P_{ss'}(a) \max_{a'} Q(s',a') \qquad (1)$$

- We can show that $f(\mathbf{Q})$ is a contraction.
- We want to find $\mathbf{Q}^*$ that satisfies $\mathbf{Q}^* = f(\mathbf{Q}^*)$.
- But because we do not know the environment, we can't observe $f(\mathbf{Q})$ directly for any $\mathbf{Q}$.

- But we can put $\mathbf{Q}$ into black box and observe

$$\begin{aligned} Y =& r(s,a) + \beta \max_{a'} Q(S', a') \\ =& f(\mathbf{Q})(s,a) + D(s,a) \end{aligned}$$

where

$$D(s,a) \triangleq \beta \left[\max_{a'} Q(S', a) - \sum_{s'} P_{ss'}(a) \max_{a'} Q(s', a') \right]$$

- D(s,a) is zero mean (conditioned on past).
- Thus, finding $\mathbf{Q}^*$ (with unknown environment) is a SA problem!
- Can therefore find $\mathbf{Q}^*$ by using the Robbins-Monro algorithm.

Robbins-Monro: replacing $\mathbf{x}$ with $\mathbf{Q}$ gives:

$$\begin{aligned} Q_{n+1}(s,a) =& Q_n(s,a) + \alpha_n \left[f(Q_n)(s,a) + D_n(s,a) - Q_n(s,a) \right] \\ =& Q_n(s,a) + \alpha_n \left[r(s,a) + \beta \max_{a'} Q_n(S',a') - Q_n(s,a) \right] \end{aligned}$$

which is the Q-learning algorithm. Therefore, from SA result:

- If $\sum_n \alpha_n = \infty$, $\sum_n \alpha_n^2 < \infty$ w.p.1
- Then $Q_n(s,a) \to Q^*(s,a)$ w.p.1 for all s, a.

Notes

- Using ϵ-greedy policy, at time n update $Q(s, a)$ only for $(s, a) = (S_n, A_n)$. This corresponds to the choice of gains:

$$\alpha_n(s; a) > 0 \text{ iff } (s; a) = (S_n, A_n)$$

- A common choice is: if $(s; a) = (S_n, A_n)$, then $\alpha_n(s; a) = 1/N_n(s, a)$, where $N_n(s, a)$ is the number of times the process has visited (s, a) up to time n.