Reinforcement Learning: Value and Policy Iteration

Manuela Veloso

Carnegie Mellon University
Computer Science Department

15-381 - Fall 2001
Reinforcement Learning Problem

Agent

Environment

State Reward Action

\[ r_0 + \gamma r_1 + \gamma^2 r_2 + \ldots, \text{ where } 0 \leq \gamma < 1 \]

Goal: Learn to choose actions that maximize

Veloso, Carnegie Mellon
Learning for Deterministic Worlds

For each $s, a$ initialize table entry $\hat{Q}(s, a) \leftarrow 0$

Observe current state $s$

Do forever:

- Select an action $a$ and execute it
- Receive immediate reward $r$
- Observe the new state $s'$
- Update the table entry for $\hat{Q}(s, a)$ as follows:

$$\hat{Q}(s, a) \leftarrow r + \gamma \max_{a'} \hat{Q}(s', a')$$

- $s \leftarrow s'$
Example - Deterministic

How many possible policies are there in this 3-state, 2-action deterministic world?

A robot starts in the state Mild. It moves for 4 steps choosing actions West, East, East, West. The initial values of its Q-table are 0 and the discount factor is $\gamma = 0.5$.

<table>
<thead>
<tr>
<th>Initial State: MILD</th>
<th>Action: West</th>
<th>Action: East</th>
<th>Action: East</th>
<th>Action: West</th>
</tr>
</thead>
<tbody>
<tr>
<td>EAST</td>
<td>HOT</td>
<td>MILD</td>
<td>COLD</td>
<td>MILD</td>
</tr>
<tr>
<td>EAST</td>
<td>10 West</td>
<td>10 East</td>
<td>-10 West</td>
<td>East</td>
</tr>
</tbody>
</table>

Initial State: MILD
Action: West
New State: HOT
New State: MILD
New State: COLD
New State: MILD

<table>
<thead>
<tr>
<th>Initial State: MILD</th>
<th>Action: West</th>
<th>Action: East</th>
<th>Action: East</th>
<th>Action: West</th>
</tr>
</thead>
<tbody>
<tr>
<td>EAST</td>
<td>HOT</td>
<td>MILD</td>
<td>COLD</td>
<td>MILD</td>
</tr>
<tr>
<td>EAST</td>
<td>0 West</td>
<td>0 East</td>
<td>0 West</td>
<td>0 West</td>
</tr>
<tr>
<td>MILD</td>
<td>0 West</td>
<td>0 East</td>
<td>10 East</td>
<td>0 East</td>
</tr>
<tr>
<td>COLD</td>
<td>0 West</td>
<td>0 East</td>
<td>0 East</td>
<td>0 East</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Initial State: MILD</th>
<th>Action: West</th>
<th>Action: East</th>
<th>Action: East</th>
<th>Action: West</th>
</tr>
</thead>
<tbody>
<tr>
<td>EAST</td>
<td>HOT</td>
<td>MILD</td>
<td>COLD</td>
<td>MILD</td>
</tr>
<tr>
<td>EAST</td>
<td>5 West</td>
<td>0 East</td>
<td>5 East</td>
<td>0 West</td>
</tr>
<tr>
<td>MILD</td>
<td>0 West</td>
<td>10 East</td>
<td>0 East</td>
<td>10 East</td>
</tr>
<tr>
<td>COLD</td>
<td>0 West</td>
<td>0 East</td>
<td>0 East</td>
<td>0 East</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Initial State: MILD</th>
<th>Action: West</th>
<th>Action: East</th>
<th>Action: East</th>
<th>Action: West</th>
</tr>
</thead>
<tbody>
<tr>
<td>EAST</td>
<td>HOT</td>
<td>MILD</td>
<td>COLD</td>
<td>MILD</td>
</tr>
<tr>
<td>EAST</td>
<td>5 West</td>
<td>0 East</td>
<td>5 East</td>
<td>0 West</td>
</tr>
<tr>
<td>MILD</td>
<td>0 West</td>
<td>0 East</td>
<td>0 East</td>
<td>0 East</td>
</tr>
<tr>
<td>COLD</td>
<td>0 West</td>
<td>0 East</td>
<td>0 East</td>
<td>0 East</td>
</tr>
</tbody>
</table>
Why is the policy $\pi(s) = \text{West}, \text{for all states}$, better than the policy $\pi(s) = \text{East}, \text{for all states}$?

- $\pi_1(s) = \text{West}, \text{for all states}, \gamma = 0.5$
  
  $$V^{\pi_1}(\text{HOT}) = 10 + \gamma V^{\pi_1}(\text{HOT}) = 20.$$  

- $\pi_2(s) = \text{East}, \text{for all states}, \gamma = 0.5$
  
  $- V^{\pi_2}(\text{COLD}) = -10 + \gamma V^{\pi_2}(\text{COLD}) = -20,$
  
  $- V^{\pi_2}(\text{MILD}) = 0 + \gamma V^{\pi_2}(\text{COLD}) = -10,$
  
  $- V^{\pi_2}(\text{HOT}) = 0 + \gamma V^{\pi_2}(\text{MILD}) = -5.$
Another Deterministic Example

$r(s, a)$ values

$Q(s, a)$ values

$V^*(s)$ values

One optimal policy

Veloso, Carnegie Mellon
Nondeterministic Case

What if reward and next state are non-deterministic?

We redefine $V, Q$ by taking expected values

\[
V^\pi(s) \equiv E[r_t + \gamma r_{t+1} + \gamma^2 r_{t+2} + \ldots]
\]

\[
\equiv E[\sum_{i=0}^{\infty} \gamma^i r_{t+i}]
\]

\[
Q(s, a) \equiv E[r(s, a) + \gamma V^*(\delta(s, a))]
\]
Nondeterministic Case

$Q$ learning generalizes to nondeterministic worlds

Alter training rule to

$$
\hat{Q}_n(s, a) \leftarrow (1 - \alpha_n)\hat{Q}_{n-1}(s, a) + \\
\alpha_n[r + \gamma \max_{a'}\hat{Q}_{n-1}(s', a')],
$$

where $\alpha_n = \frac{1}{1 + \text{visits}_n(s,a)}$, and $s' = \delta(s, a)$.

$\hat{Q}$ still converges to $Q^*$ (Watkins and Dayan, 1992)
Nondeterministic Example

S1
Unemployed

S2
Industry

S3
Grad School

S4
Academia

REWARD

<table>
<thead>
<tr>
<th></th>
<th>S1</th>
<th>S2</th>
<th>S3</th>
<th>S4</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>0</td>
<td>100</td>
<td>0</td>
<td>10</td>
</tr>
</tbody>
</table>
Nondeterministic Example

\( \pi^*(s) = D \), for any \( s = S_1, S_2, S_3, \) and \( S_4, \gamma = 0.9. \)

\[
V^*(S_2) = r(S_2, D) + 0.9 \left( 1.0 \ V^*(S_2) \right) \\
V^*(S_2) = 100 + 0.9 \ V^*(S_2) \\
V^*(S_2) = 1000. 
\]

\[
V^*(S_1) = r(S_1, D) + 0.9 \left( 1.0 \ V^*(S_2) \right) \\
V^*(S_1) = 0 + 0.9 \times 1000 \\
V^*(S_1) = 900. 
\]

\[
V^*(S_3) = r(S_3, D) + 0.9 \left( 0.9 \ V^*(S_2) + 0.1 \ V^*(S_3) \right) \\
V^*(S_3) = 0 + 0.9 \left( 0.9 \times 1000 + 0.1 \ V^*(S_3) \right) \\
V^*(S_3) = \frac{81000}{91}. 
\]

\[
V^*(S_4) = r(S_4, D) + 0.9 \left( 0.9 \ V^*(S_2) + 0.1 \ V^*(S_4) \right) \\
V^*(S_4) = 40 + 0.9 \left( 0.9 \times 1000 + 0.1 \ V^*(S_4) \right) \\
V^*(S_4) = \frac{85000}{91}. 
\]
What is the $Q$-value, $Q(S2,R)$?

\[
Q(S2,R) = r(S2,R) + 0.9 \left( 0.9 \, V^*(S1) + 0.1 \, V^*(S2) \right)
\]

\[
Q(S2,R) = 100 + 0.9 \left( 0.9 \times 900 + 0.1 \times 1000 \right)
\]

\[
Q(S2,R) = 100 + 0.9 \, (810 + 100)
\]

\[
Q(S2,R) = 100 + 0.9 \times 910
\]

\[
Q(S2,R) = 919.
\]
Markov Decision Processes

- Finite set of states, \( s_1, \ldots, s_n \)
- Finite set of actions, \( a_1, \ldots, a_m \)
- Probabilistic state, action transitions:
  \[
  p^k_{ij} = \text{prob} \ (\text{next} = s_j \ | \ \text{current} = s_i \ \text{and take action} \ a_k)
  \]
- Reward for each state and action.
- Process:
  - Start in state \( s_i \)
  - Choose action \( a_k \in A \)
  - Receive immediate reward \( r_i(s_i, a_k) \)
  - Change to state \( s_j \) with probability \( p^k_{ij} \).
  - Discount future rewards
Solving an MDP

- A policy is a mapping from states to actions.

- Optimal policy - for every state, there is no other action that gets a higher sum of discounted future rewards.

- For every MDP there exists an optimal policy.

- Solving an MDP is finding an optimal policy.

- A specific policy converts an MDP into a plain Markov system with rewards.
Policy Iteration

- Start with some policy \( \pi_0(s_i) \).
- Such policy transforms the MDP into a plain Markov system with rewards.
- Compute the values of the states according to current policy.
- Update policy:

\[
\pi_1(s_i) = \arg\max_a \{ r_i + \gamma \sum_j p_{ij} V^{\pi_0}(s_j) \}
\]

- Keep computing
- Stop when \( \pi_{k+1} = \pi_k \).
Value Iteration

- \( V^*(s_i) \) = expected discounted future rewards, if we start from \( s_i \) and we follow the optimal policy.

- Compute \( V^* \) with value iteration:
  - \( V^k(s_i) \) = maximum possible future sum of rewards starting from state \( s_i \) for \( k \) steps.

- Bellman’s Equation:

  \[
  V^{n+1}(s_i) = \max_k \{ r_i + \gamma \sum_{j=1}^{N} p_{ij}^k V^n(s_j) \}
  \]

- Dynamic programming
Summary

- Q-learning
- Markov decision processes
- Value, policy iteration