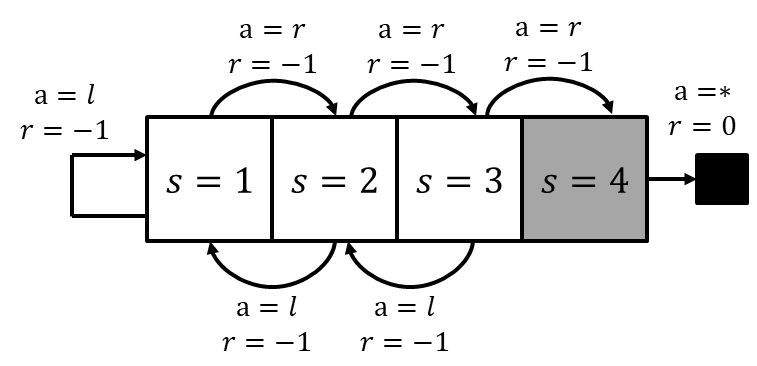
**A Complete Toy Example**

1. Consider the following MDP. Environment is deterministic. In each state, there are two possible actions a∈{l,r}, where l corresponds to moving left, and r corresponds to moving right. Each movement incurs a reward of r=-1. State s=4 is the goal state: taking any action from s=4 results in reward of r=0 and ends the episode, hence V(4)≡0,Q(4,a)≡0 for any action a.



Assume . All value functions are initialized to 0.

A. Use Policy Iteration, Value Iteration to derive optimal policy.

B. Consider 8 consecutive episodes in the form of (s,a,r):

1. EP1:
2. EP2:
3. EP3:
4. EP4:
5. EP5:
6. EP6:
7. EP7:
8. EP8:

Derive the following (only show the changed parts):

1. State value functions after TD learning.
2. State-action value functions (Q Value Functions) after Sarsa, and the resulting policy.
3. State-action value functions (Q Value Functions) after Q learning, and the resulting policy.

ANS:

**A. Use Policy Iteration, Value Iteration to derive optimal policy.**

**1.1 Policy Evaluation**

Bellman Exp Equation for uniform random policy:

Solution:

**1.2 Policy Improvement**

Plug in values from PE to get new policy

**2.1 Policy Evaluation**

Solution:

**2.2 Policy Improvement**

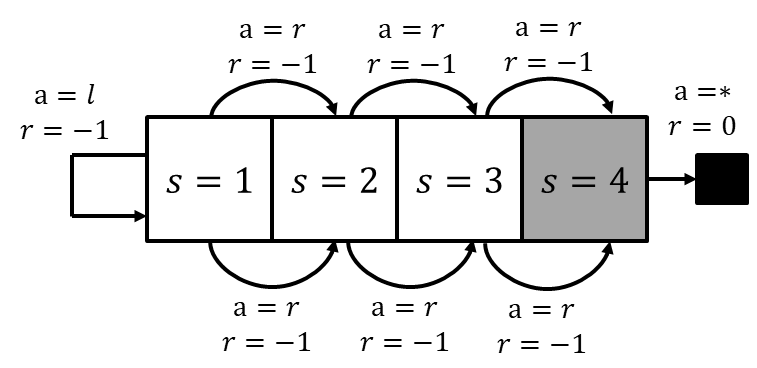
Plug in values from PE to get new policy

Policy is now stable (Value functions happen to have also converged.)

**A. Use Value Iteration to derive optimal policy.**

Bellman Opt Equation:

Solution: (There is no analytical method for solving nonlinear equations with operator. In this case the solution is easy to see by observation, but in general you need to use iterative solving method.)



**B.1. State value functions after TD learning.**

TD update equation:

After EP1:

After EP2:

After EP3:

After EP4:

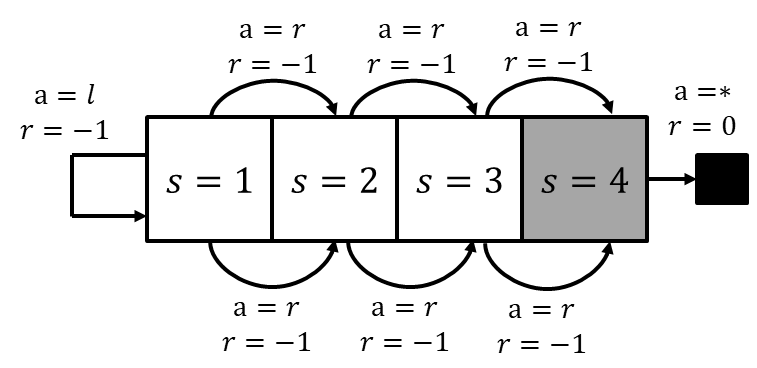
After EP5:

After EP6:

After EP7:

After EP8:

|  |  |  |  |
| --- | --- | --- | --- |
| TD |  |  |  |
| Init |  |  |  |
| After EP1 |  |  |  |
| After EP2 |  |  |  |
| After EP3 |  |  |  |
| After EP4 |  |  |  |
| After EP5 |  |  |  |
| After EP6 |  |  |  |
| After EP7 |  |  |  |
| After EP8 |  |  |  |



**My comments:** TD failed to converge. When moving left, bootstraps off , bootstraps off , and so on. The sequence of TD updates cause all value functions to be increasingly negative. Subsequently, moving right helps to set to the correct value of , but then the episode ends immediately, so and do not have a chance to bootstrap off the new . If the episode does not end immediately, but goes back to the left again, then and will have a chance to bootstrap off the new , and they will converge to the correct values.

TD cannot be used to learn a policy in the model-free case.

**B.2. State-action value functions after Sarsa.**

Sarsa update equation:

EP1:

EP2:

EP3:

EP4:

1. (bootstraps off )

EP5:

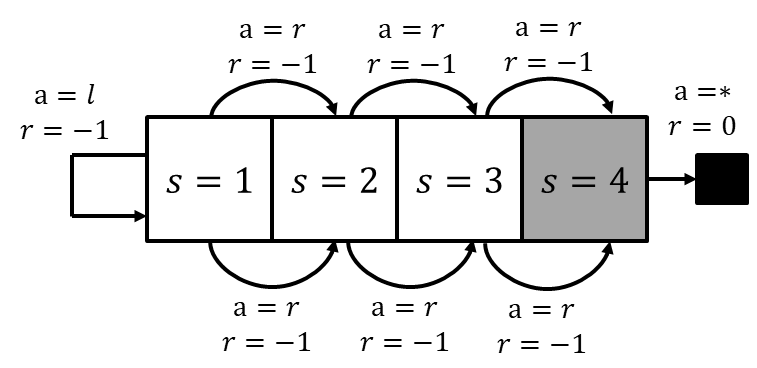
1. (bootstraps off )

EP6:

1. (bootstraps off )

EP7:

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  |  |  |  |  |  |  |
| Init |  |  |  |  |  |  |
| After EP1 |  |  |  |  |  |  |
| After EP2 |  |  |  |  |  |  |
| After EP3 |  |  |  |  |  |  |
| After EP4 |  |  |  |  |  |  |
| After EP5 |  |  |  |  |  |  |
| After EP6 |  |  |  |  |  |  |
| After EP7 |  |  |  |  |  |  |
| After EP8 |  |  |  |  |  |  |



**My comments:** Sarsa converges. State-action value functions for moving right look reasonable:. But State-action value functions for moving left look unreasonable: . Sine the only trajectories with move left actions are , the Q values are updated based on only this trajectory (on-policy), i.e., from state taking action left, it can only take the above trajectory, and reach the goal in 6 steps, hence . Even though the Q values are inaccurate and too pessimistic, the greedy policy is still optimal:

**B.3. State-action value functions after QL.**

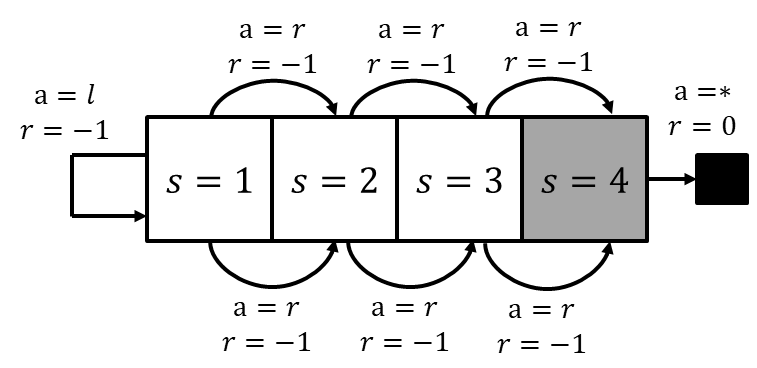
Q learning update equation:

After EP1:

After EP2:

After EP3:

**My comments:** After first 3 episodes, . This is reasonable for , but not reasonable for , as agent cannot reach goal in one step from states or . This is because all value functions bootstrap off the optimistic initial estimate of for all , which is wrong and not yet corrected since agent has never taken the left action.



EP4:

1. (bootstraps off =)
2. (bootstraps off =)

EP5:

1. (bootstraps off ,.)
2. (bootstraps off ,.)
3. (bootstraps off ,.)

EP6:

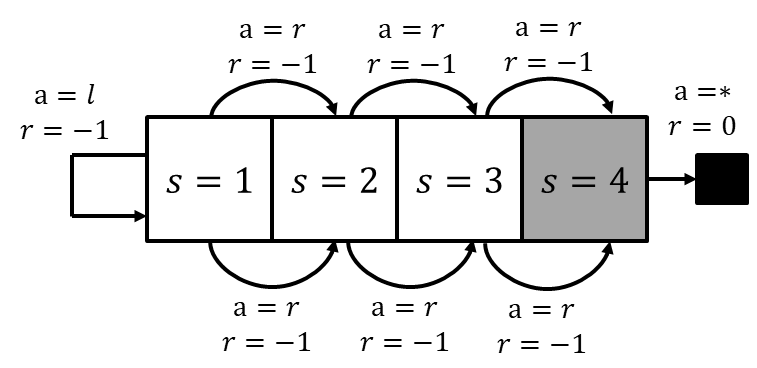
1. (bootstraps off ,.)
2. (bootstraps off ,.)
3. (bootstraps off ,.)
4. (bootstraps off ,.)

EP7:

1. (bootstraps off ,.)
2. (bootstraps off ,.)
3. (bootstraps off ,.)

EP8:

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Q Learning |  |  |  |  |  |  |
| Init |  |  |  |  |  |  |
| After EP1 |  |  |  |  |  |  |
| After EP2 |  |  |  |  |  |  |
| After EP3 |  |  |  |  |  |  |
| After EP4 |  |  |  |  |  |  |
| After EP5 |  |  |  |  |  |  |
| After EP6 |  |  |  |  |  |  |
| After EP7 |  |  |  |  |  |  |
| After EP8 |  |  |  |  |  |  |



**My comments:** QL converges. All State-action value functions look reasonable:.

: If agent moves left in state , then follow optimal policy of moving right, then it takes 4 steps to reach goal .

: If agent moves left in state , then follow optimal policy of moving right, then it takes 4 steps to reach goal .

: If agent moves left in state , then follow optimal policy of moving right, then it takes 3 steps to reach goal .

So QL is smarter than Sarsa: since it is off-policy, agent can learn the correct Q value functions that correspond to trajectories that it has never actually experienced, e.g., agent never experienced the trajectory , but agent’s Q value functions are updated based on that trajectory (by taking ).

Q values learned by QL are accurate, and obviously the greedy policy is optimal: