Reinforcement Learning IV

Reinforcement Learning Roadmap

Knowledge of **Environment**

1 Core concepts in reinforcement learning

Actions, Rewards, Value, Environments, and Policies

Perfect knowledge
Known Markov Decision Process

2 Markov decision processes

…and Markov chains and Markov reward processes

3 Dynamic Programming

How do we find optimal policies?
(Bellman equations)

No knowledge
Must learn from experience

4 Monte Carlo Control

How do we estimate our value functions?
How do we use the value functions to choose actions?
How do we learn optimal policies from experience?

Roadmap to optimal policies

If we assume a **fully known MDP environment**, how do we…
(Markov Decision Process)

1. Evaluate the returns a policy will yield? **Policy evaluation**
2. Find a **better** policy? **Policy improvement**
3. Find the **best** policy? **Policy iteration**
4. Find the best policy **faster**? **Value iteration**

Dynamic Programming

What if we don't have a fully known MDP? **Monte Carlo Methods**

1. Policy Evaluation

Evaluate the returns a policy will yield

Input: policy $\pi(a|s)$

Output: value function $v_\pi(s)$

(unknown)

1. Select a policy function to evaluate (estimate the value function)
2. Start with a guess of the value function, v_0 (often all zeros)
3. **Iteratively** apply the Bellman Expectation Equation to "backup" the values until they converge on the actual value function for the policy, v_π

$$v_0 \rightarrow v_1 \rightarrow \cdots \rightarrow v_\pi$$

PREVIOUSLY

Adapted from David Silver, 2015

Monte Carlo Policy Evaluation

For **state** values

Evaluate the returns a policy will yield

Input: policy $\pi(a|s)$
Output: state value $v_\pi(s)$

1. Select a policy function to evaluate (estimate the value function)
2. Start with a guess of the value function, v_0 (often all zeros)
3. Estimate the value function through experience by iterating:
   - A. Generate an episode (take actions until a terminal state)
   - B. Save the returns following the first occurrence of each state
   - C. Assign $\text{AVG}(\text{Returns}(s)) \rightarrow \hat{v}_\pi(s)$

Sutton and Barto, 1998

Monte Carlo Policy Evaluation

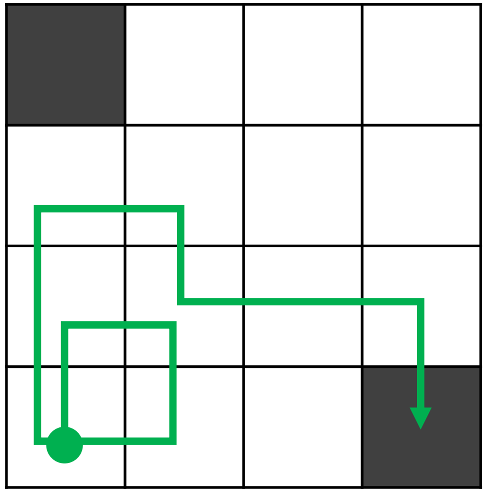
For **state** values

"First Visit"

For each state, we store the running returns seen **after** the first visit to that state

Episode 1

Total Reward: -11



Episode 1 **returns** after the first visit of each state

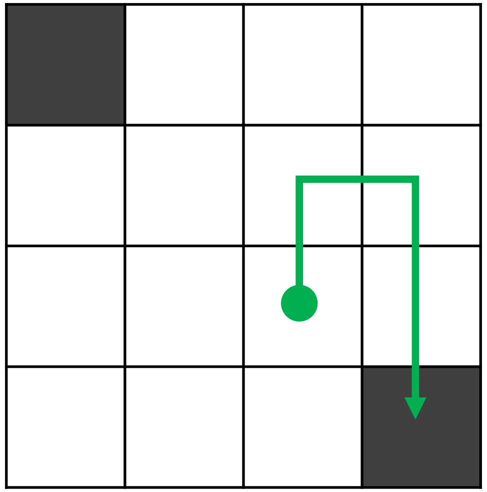
$G^{(1)}$

-5	-4		
-10	-9	-2	-1
-11	-8		

Discount rate: $\gamma = 1$

Episode 2

Total Reward: -4



Episode 2 **returns** from the first visit of each state

$G^{(2)}$

		-3	-2
		-4	-1

Discount rate: $\gamma = 1$

$v_0(s)$

0	0	0	0
0	0	0	0
0	0	0	0
0	0	0	0

The value function is the **running average** of the returns after the visit to that state, averaged over episodes

(only average over episodes when state is visited)

$v_1(s)$

0	0	0	0
-5	-4	0	0
-10	-9	-2	-1
-11	-8	0	0

v_1 is just the first visit returns, $G^{(1)}$

v_2 is the average first visit returns, $G^{(1)}$ and $G^{(2)}$, for those states visited

$v_2(s)$

0	0	0	0
-5	-4	-3	-2
-10	-9	-3	-1
-11	-8	0	0

State vs action value

The **state value function** doesn't tell us directly about actions

If we don't have a model, to pick a policy we need **action values**

State vs action value

Greedy policy improvement over $v(s)$ **requires a model of the MDP**

$$\pi'(s) = \underset{a}{\operatorname{argmax}}\, \underset{?}{R_{t+1}} + \underset{?}{p(s', r|s, a)}\, v_\pi(s')$$

Greedy policy improvement over $q_\pi(s, a)$ **requires no MDP knowledge**

$$\pi'(s) = \underset{a}{\operatorname{argmax}}\, q_\pi(s, a)$$

And the two value functions are related: $v_\pi(s) = \sum_a \pi(a|s) q_\pi(s, a)$

David Silver, UCL, 2015

Monte Carlo Policy Evaluation

For **action** values

Input: policy $\pi(a|s)$

Output: **action value** $q_\pi(s,a)$

Evaluate the returns a policy will yield

1. Select a policy function to evaluate (estimate its value function)
2. Start with a guess of the action value function, q_0 (often all zeros)
3. Repeat forever:
   - A. Generate an episode (take actions until a terminal state)
   - B. Save returns following first occurrence of each state **& action**
   - C. Assign $\text{AVG}(\text{Returns}(\text{s}, \text{a})) \rightarrow \hat{q}_\pi(s,a)$

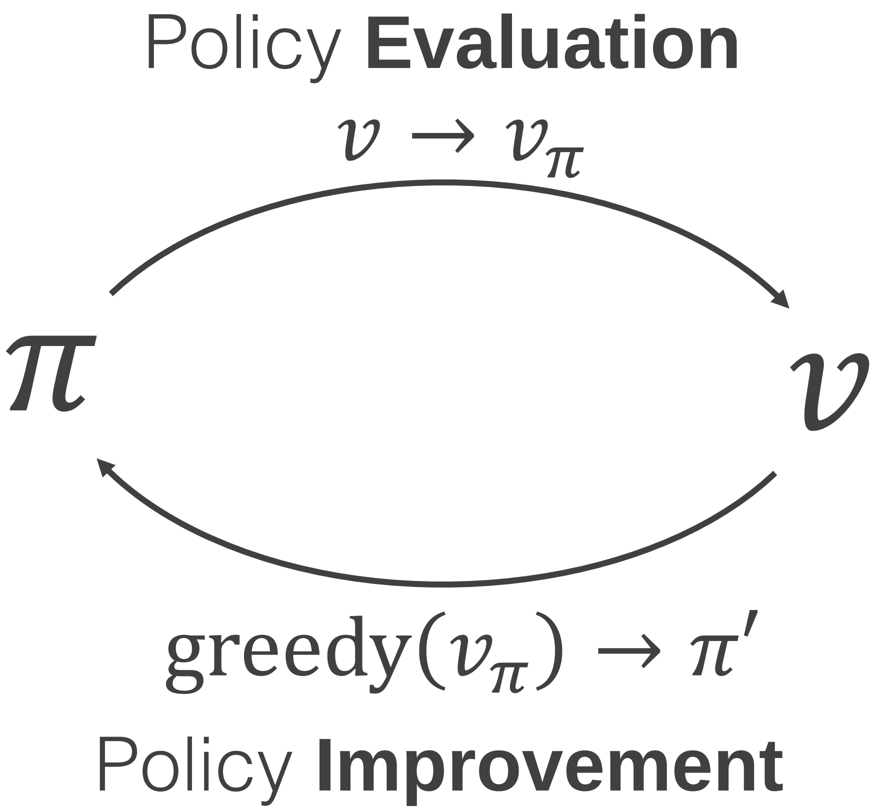
Sutton and Barto, 1998

3. Policy Iteration

Find the **best** policy

Input: policy $\pi(a|s)$

Output: **best** policy $\pi^*(a|s)$



1. **Policy Evaluation**: estimate v_π
   Iterative policy evaluation
   Note: This is VERY slow
2. **Policy Improvement**: generate $\pi' \geq \pi$
   Greedy policy improvement
3. Iterate 1 and 2 until convergence

PREVIOUSLY

Adapted from David Silver, 2015 and Sutton and Barto, 1998

Monte Carlo Control

Find the **best** policy

Policy **Evaluation**

$q \rightarrow q_\pi$

π $\qquad$ q

$\text{greedy}(q_\pi) \rightarrow \pi'$

Policy **Improvement**

1. **Policy Evaluation**: estimate q_π
   Monte Carlo action policy evaluation
2. **Policy Improvement**: generate $\pi' \geq \pi$
   Greedy policy improvement
3. Iterate 1 and 2 until convergence

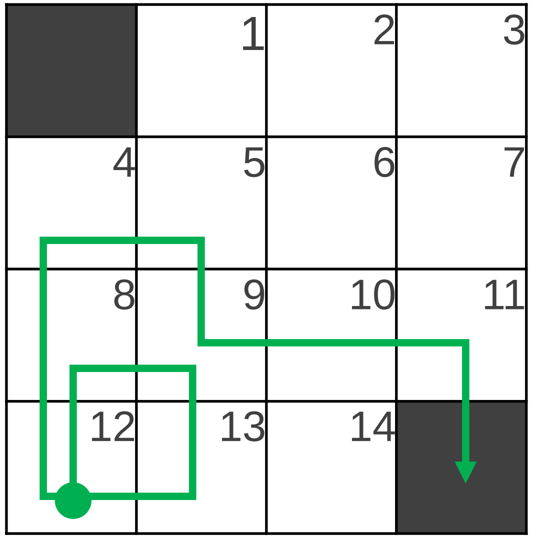
Sutton and Barto, 1998

Monte Carlo Control

“First Visit” (of state AND action) is recorded

Episode 1

Total Reward: -11



← State labels

1 MC Policy Evaluation

Episode 1 **returns** after the first visit of each state

-5	**-4**		
-10	**-9**	**-2**	**-1**
-11	**-8**		

Discount rate: $\gamma = 1$

$q_\pi(s, a)$

Action (a): ↑ → ← ↓

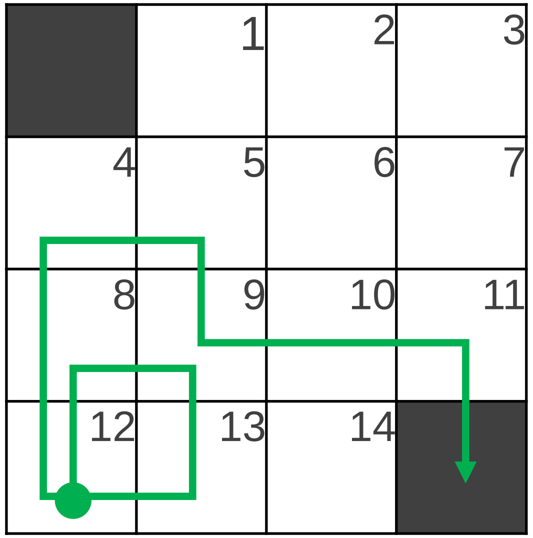
State (s)	↑	→	←	↓
1				
2				
3				
4				
5				
6				
7				
8				
9				
10				
11				
12				
13				
14				

Monte Carlo Control

"First Visit" (of state AND action) is recorded

Episode 1

Total Reward: -11



State labels

Episode 1 **returns** after the first visit of each state

	1	2	3
4 **-5**	5 **-4**	6	7
8 **-10**	9 **-9**	10 **-2**	11 **-1**
12 **-11**	13 **-8**	14	

Discount rate: $\gamma = 1$

1. MC Policy Evaluation
2. MC Policy Improvement

$$\pi'(s) = \underset{a}{\operatorname{argmax}}\, q_\pi(s,a)$$

Typically this is set to be ϵ-greedy to better learn $q_\pi(s,a)$

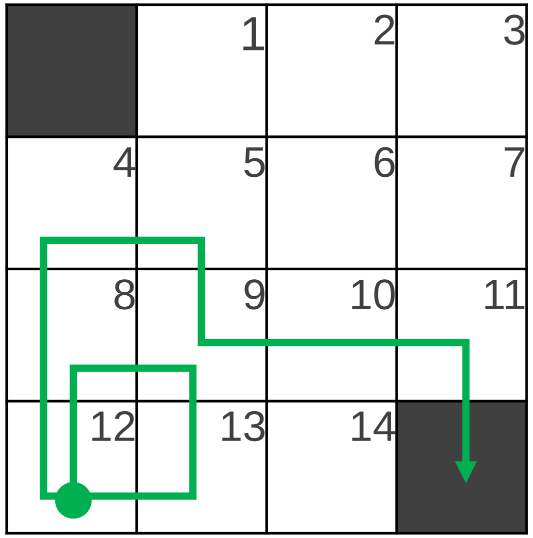
$q_\pi(s,a)$

Invalid action

Action (a): ↑ → ← ↓

State (s)	↑	→	←	↓
1				
2				
3				
4		**-5**		
5				**-4**
6				
7				
8	**-6**	**-10**		
9		**-3**		**-9**
10		**-2**		
11				**-1**
12	**-11**			
13			**-8**	
14				

Episode 1



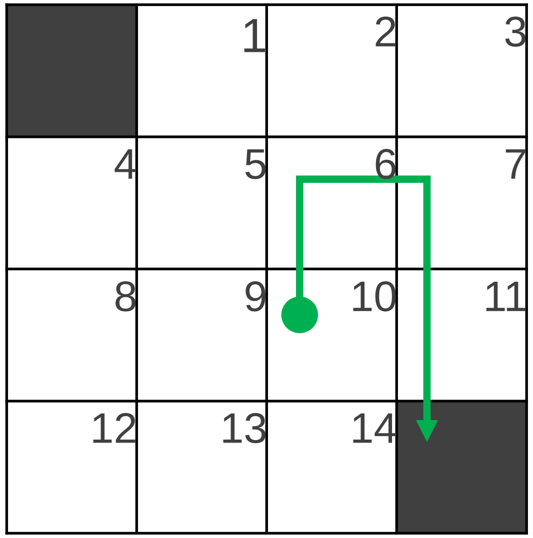
	1	2	3
4: -5	5: -4	6	7
8: -10	9: -9	10: -2	11: -1
12: -11	13: -8	14	

Returns from episode 1

$q_{\pi_1}(s,a)$

State (s) \ Action (a)	↑	→	←	↓
1				
2				
3				
4		-5		
5				-4
6				
7				
8	-6	-10		
9		-3		-9
10		-2		
11				-1
12	-11			
13			-8	
14				

Episode 2



	1	2	3
4	5	6: -3	7: -2
8	9	10: -4	11: -1
12	13	14	

Returns from episode 2

$q_{\pi_2}(s,a)$

State (s) \ Action (a)	↑	→	←	↓
1				
2				
3				
4		-5		
5				-4
6		-3		
7				-2
8	-6	-10		
9		-3		-9
10	-4	-2		
11				-1
12	-11			
13			-8	
14				

$q_{\pi^*}(s,a)$

State (s) \ Action (a):	↑	→	←	↓
1		-3	-1	-3
2		-4	-2	-3
3			-3	-3
4	-1	-3		-3
5	-2	-4	-2	-4
6	-3	-3	-3	-3
7	-4		-4	-2
8	-2	-4		-4
9	-3	-3	-3	-3
10	-4	-2	-4	-2
11	-3		-3	-1
12	-3	-3		
13	-4	-2	-4	
14	-3	-1	-3	

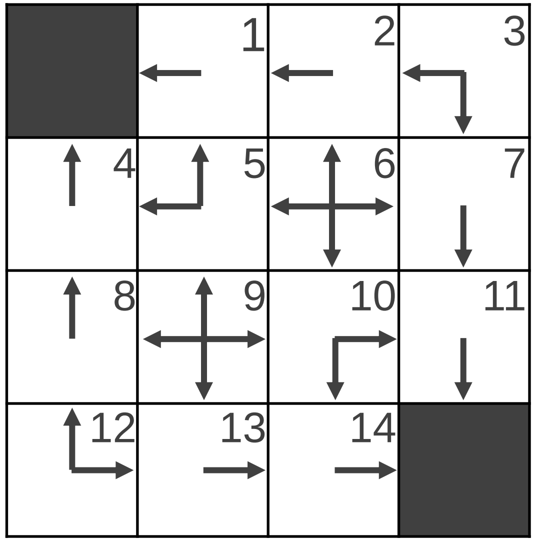
If we're in state 4, take the up action

If we know the optimal action value function, we also have our optimal policy

$v_{\pi^*}(s)$

0	-1 (1)	-2 (2)	-3 (3)
-1 (4)	-2 (5)	-3 (6)	-2 (7)
-2 (8)	-3 (9)	-2 (10)	-1 (11)
-3 (12)	-2 (13)	-1 (14)	0

$\pi^*(s)$



Extensions

Monte Carlo methods require that we finish each episode before updating
Solution: **Temporal Difference** (TD) methods

What if we want to learn about one policy while following or observing another?
(e.g. evaluate a greedy policy while exploring the state space)
Solution: **Off-policy learning** instead of on-policy learning

What if our state space has too many states that we can't build a table of values?
Solution: **Value function approximation** (involving supervised learning techniques)

How can we simulate what the environment might output for next states and rewards?
Solution: **Model-based learning**: simulate the environment and plan ahead

Reinforcement Learning Roadmap

Knowledge of **Environment**

1 Core concepts in reinforcement learning

Actions, Rewards, Value, Environments, and Policies

Perfect knowledge
Known Markov Decision Process

2 Markov decision processes

…and Markov chains and Markov reward processes

3 Dynamic Programming

How do we find optimal policies?
(Bellman equations)

No knowledge
Must learn from experience

4 Monte Carlo Control

How do we estimate our value functions?
How do we use the value functions to choose actions?
How do we learn optimal policies from experience?