Reinforcement Learning IV

Reinforcement Learning Roadmap

Knowledge of **Environment**

Perfect knowledge
Known Markov Decision Process

No knowledge
Must learn from experience

Dynamic Programming

What's a Markov Decision Process?
How do we find optimal policies?

Monte Carlo Control

How do we estimate our value functions?
How do we use the value functions to choose actions?

Markov Decision Process

Components:

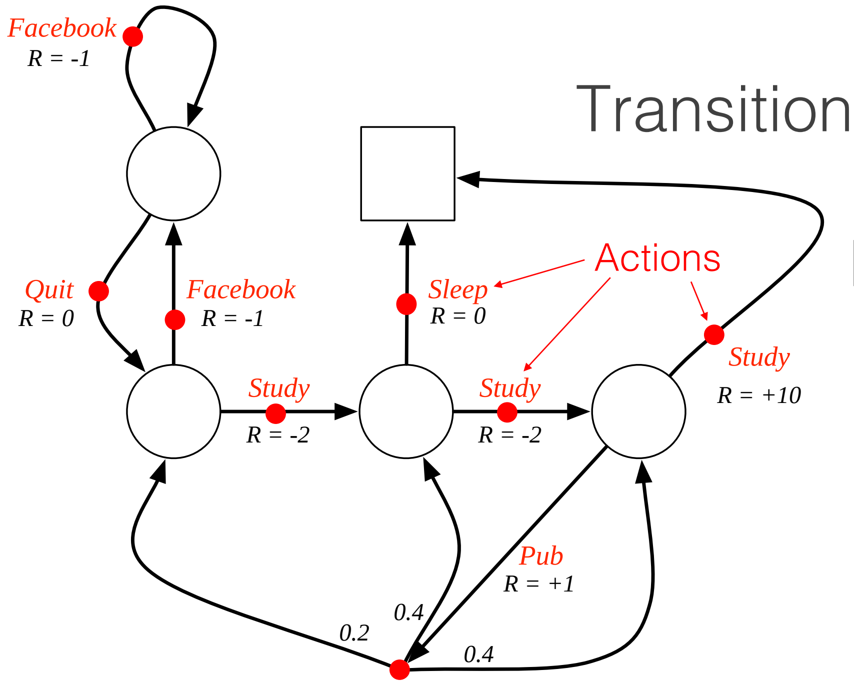
State space S

Transition probabilities, P

Rewards, R

Discount rate, γ

Actions, A



Returns (Expected future rewards)

(discount factor weights the the future)

$$G_t = R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+3} \ldots$$

Policy (how we choose actions)

(can be stochastic or deterministic)

$$\pi(a|s) = P(a|s)$$

State value function

(expected return from state s, and following policy π)

$$v_\pi(s) = E[G_t|s]$$

$$v_\pi(s) = E[R_s^a + \gamma v_\pi(s')|s]$$

Action value function

(expected return from state s, taking action a, and following policy π)

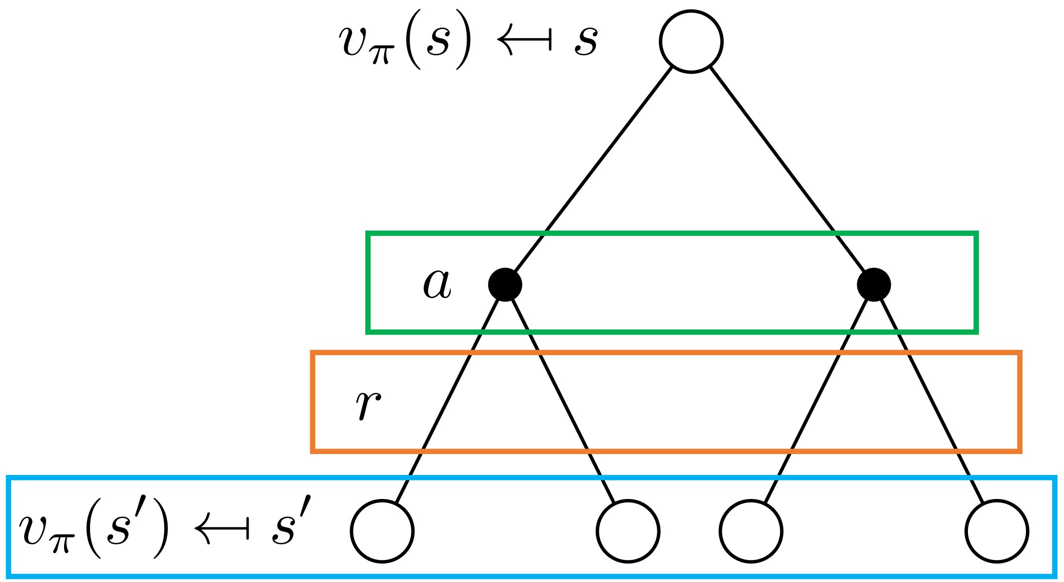
$$q_\pi(s,a) = E[G_t|s,a]$$

$$q_\pi(s,a) = E[R_s^a + \gamma q_\pi(s',a')|s,a]$$

David Silver, UCL, 2015

Bellman Expectation Equations for the state value function

(expected return from state s, and following policy π)



$$v_\pi(s) = E[G_t|s]$$

$$v_\pi(s) = E[R_s^a + \gamma v_\pi(s')|s]$$

$$R_s^a = E[r_{t+1}|S_t = s, A_t = a]$$

Expectation over the possible actions

Expectation over the rewards

(based on state and choice of action)

Expectation over the next possible states

$$v_\pi(s) = \underbrace{\sum_a \pi(a|s)}\left(\underbrace{R_s^a} + \gamma \underbrace{\sum_{s'} P_{ss'}^a v_\pi(s')}\right)$$

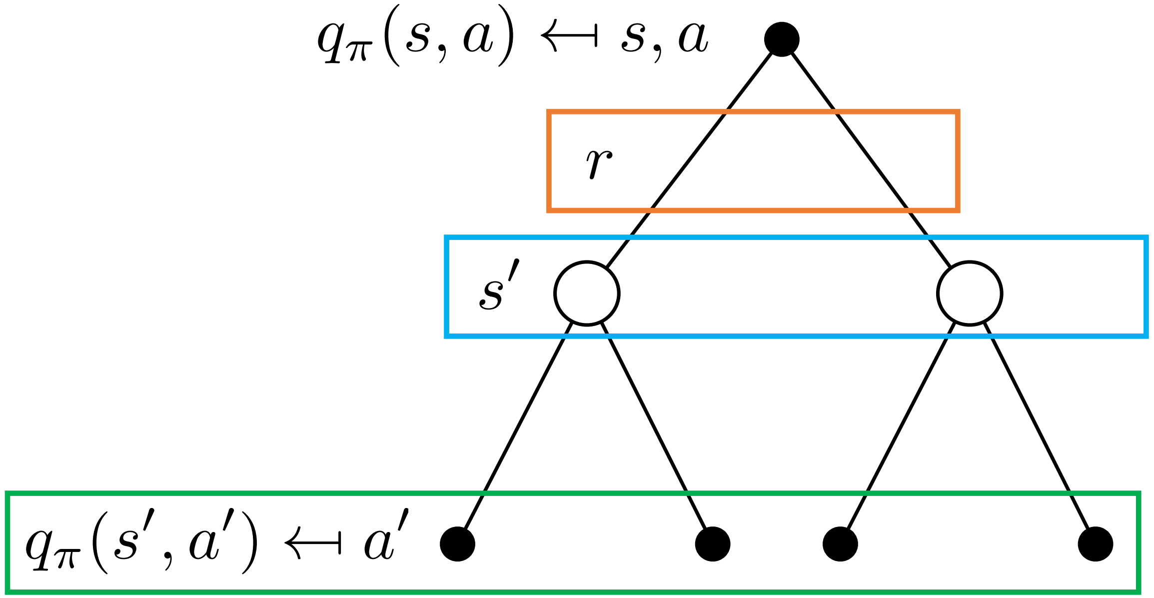
Bellman Expectation Equations for the action value function

(expected return from state s, taking action a, then following policy π)

$$q_\pi(s,a) = E[G_t | s, a]$$

$$q_\pi(s,a) = E[R_s^a + \gamma q_\pi(s', a') | s, a]$$

$$R_s^a = E[r_{t+1} | S_t = s, A_t = a]$$



Expectation over the rewards

(based on state and choice of action)

Expectation over the next possible states

Expectation over the possible actions

$$\underbrace{q_\pi(s,a) = R_s^a}_{} + \gamma \sum_{s'} P_{ss'}^a \sum_{a'} \pi(a'|s') q_\pi(s', a')$$

Roadmap to optimal policies

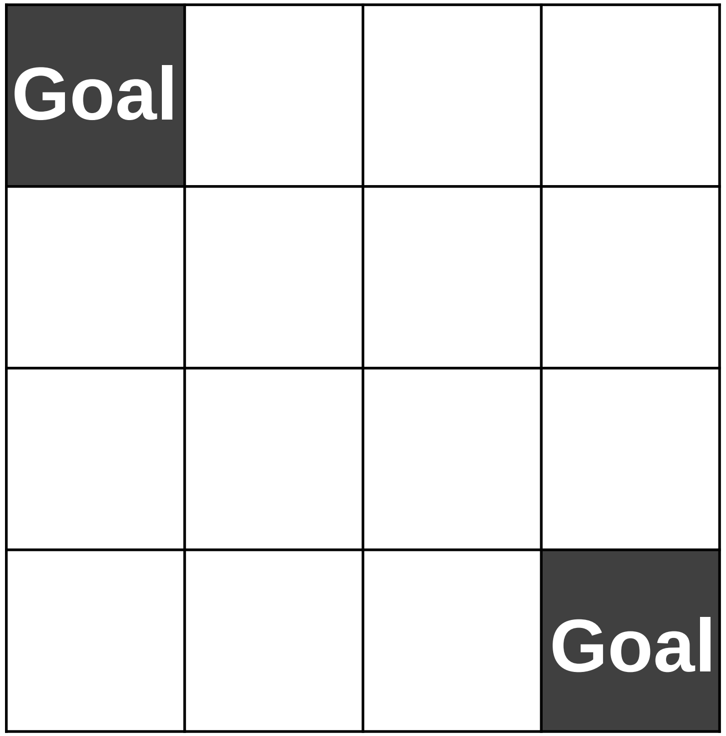
If we assume a **fully known MDP environment**, how do we…
(Markov Decision Process)

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

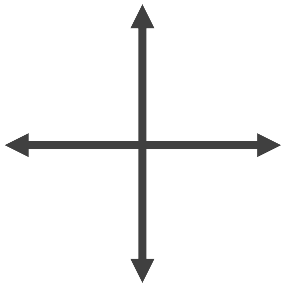
What if we don't have a fully known MDP? **Monte Carlo Methods**

Running example: Gridworld



16 states, 2 of them terminal states labeled “goal”

Valid actions:
(unless there is a wall)



Reward:

-1 for all transitions
(until the terminal state has been reached)

Note: actions that would take the agent off the board are not allowed

Sutton and Barto, 2018

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

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

Adapted from David Silver, 2015

1. Policy Evaluation

Evaluate the returns a policy will yield

Policy: $\pi(a|s) = \dfrac{1}{N_{\text{valid\_actions}}}$

Randomly go in any valid direction

Value function initialization:

$v_0(s) = 0$ (all zeros)

$v_0(s)$

(initialization)

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

We estimate the value function that corresponds to the policy: $v_\pi(s)$

1. Policy Evaluation

Evaluate the returns a policy will yield

$v_0(s)$

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

Policy: $\pi(a|s) = 1/N_{\text{valid\_actions}}$

(randomly go in any direction)

Bellman Expectation Equation:

$$v_{k+1}(s) = \sum_a \pi(a|s)\left(R_s^a + \gamma \sum_{s'} P_{ss'}^a v_k(s')\right)$$

In Gridworld: $\frac{1}{N_a}$ $\quad -1 \quad$ 1

(once you pick an action there's no uncertainty as to which state you'll transition to)

$$v_{k+1}(s) = \sum_a \frac{1}{N_a}\left(-1 + \sum_{s'} v_k(s')\right) = -1 + \sum_a \frac{1}{N_a} \sum_{s'} v_k(s') \qquad = -1 + \sum_a \frac{1}{N_a} v_k(s')$$

Each action leads to only one state, so the sum over states is not needed

Average of the value of the N_a neighboring states

1. Policy Evaluation

Evaluate the returns a policy will yield

$$v_{k+1}(s) = -1 + \sum_a \frac{1}{N_a} v_k(s')$$

$$v_1 = -1 + \sum_a \frac{1}{4} v_k(s') \quad = -1$$

$v_0(s)$

0	**0**	**0**	**0**
0	**0**	**0**	**0**
0	**0**	**0**	**0**
0	**0**	**0**	**0**

One neighborhood in $v_0(s)$

	0		
0		**0**	
	0		

$v_1(s)$

0	**-1**	**-1**	**-1**
-1	**-1**	**-1**	**-1**
-1	**-1**	**-1**	**-1**
-1	**-1**	**-1**	**0**

$v_0(s)$

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

$v_1(s)$

0	-1	-1	-1
-1	-1	-1	-1
-1	-1	-1	-1
-1	-1	-1	0

$v_2(s)$

0	-1.7	-2	-2
-1.7	-2	-2	-2
-2	-2	-2	-1.7
-2	-2	-1.7	0

$v_3(s)$

0	-2.4	-2.9	-3.0
-2.4	-2.9	-3.0	-2.9
-2.9	-3.0	-2.9	-2.4
-3.0	-2.9	-2.4	0

$v_{10}(s)$

0	-6.1	-8.4	-9.0
-6.1	-7.7	-8.4	-8.4
-8.4	-8.4	-7.7	-6.1
-9.0	-8.4	-6.1	0

$v_\infty(s) = v_\pi(s)$

0	-14	-20	-22
-14	-18	-20	-20
-20	-20	-18	-14
-22	-20	-14	0

We've found the value function (expected returns) from our random movement policy

1. Policy Evaluation

Evaluate the returns a policy will yield

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**

2. Policy Improvement

Find a **better** policy

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

Output: better policy $\pi'(a|s)$

Definition of better: has greater or equal expected return in all states:
$v_{\pi'}(s) \geq v_{\pi}(s)$ for all states

1. Select a policy function to improve
2. Evaluate the value function (our last discussion)
3. **Greedily** select a new policy, π', that chooses actions that maximize value

$$\pi'(s) = \arg\max_{a} q_{\pi}(s, a)$$

$q_{\pi}(s, a)$ = expected return from state s, taking action a, and following policy π

i.e. pick the **action** that brings us to the state with **highest value**

Adapted from David Silver, 2015

Value function:

In this case,
$q_\pi(s, \pi(s)) = v_\pi(s)$
since each action
leads to only one state

$v_0(s)$

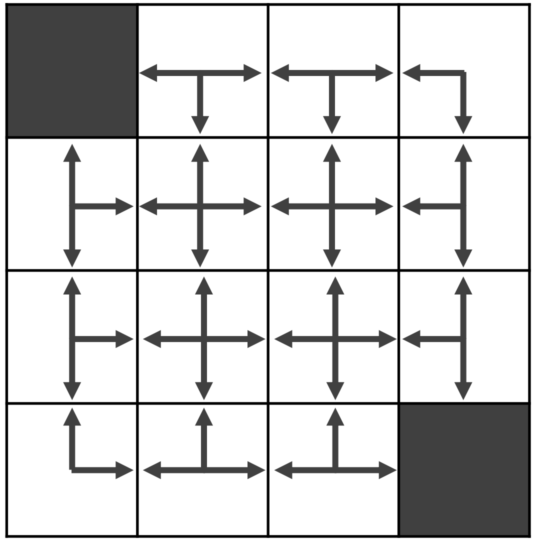
0	0	0	0
0	0	0	0
0	0	0	0
0	0	0	0

$v_\infty(s) = v_\pi(s)$

0	-14	-20	-22
-14	-18	-20	-20
-20	-20	-18	-14
-22	-20	-14	

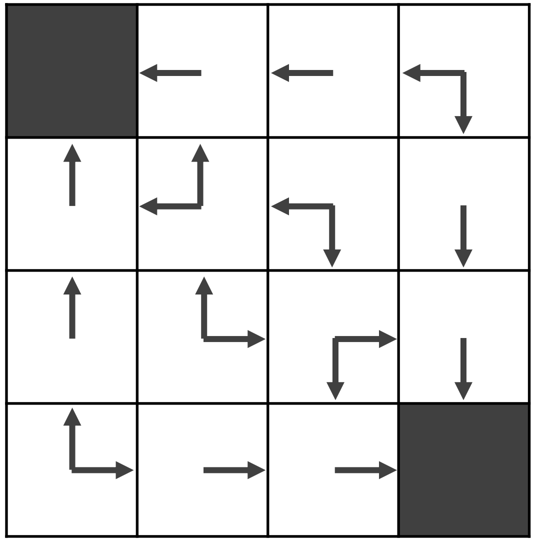
Initial policy: $\pi(s)$

$\pi(a|s) =$ randomly go in any valid direction



Improved policy
(in this case this is an optimal policy)

$\pi'(s)$



2. Policy Improvement
Find a **better** policy

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**	Dynamic Programming
2. Find a **better** policy?	**Policy improvement**	Dynamic Programming
3. Find the **best** policy?	**Policy iteration**	Dynamic Programming
4. Find the best policy **faster**?	**Value iteration**	Dynamic Programming
What if we don't have a fully known MDP?	**Monte Carlo Methods**	

3. Policy Iteration

Find the **best** policy

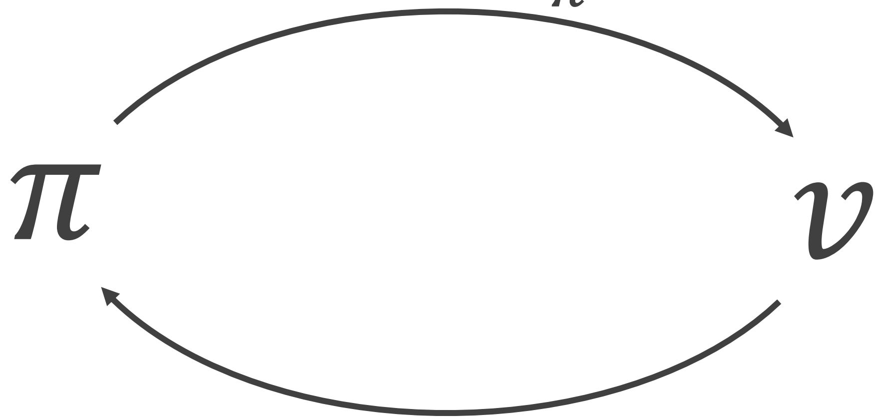
Input: policy $\pi(a|s)$

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

Best in the sense that: $v_{\pi^*}(s) \geq v_\pi(s)$ for all states and for all **policies**

Policy **Evaluation**

$v \rightarrow v_\pi$



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

Policy **Improvement**

⋮

This process will converge onto the optimal functions

$$\pi^* \rightleftarrows v^*$$

$$\pi_0 \xrightarrow{E} v_{\pi_0} \xrightarrow{I} \pi_1 \xrightarrow{E} v_{\pi_1} \rightarrow \cdots \xrightarrow{I} \pi^* \xrightarrow{E} v^*$$

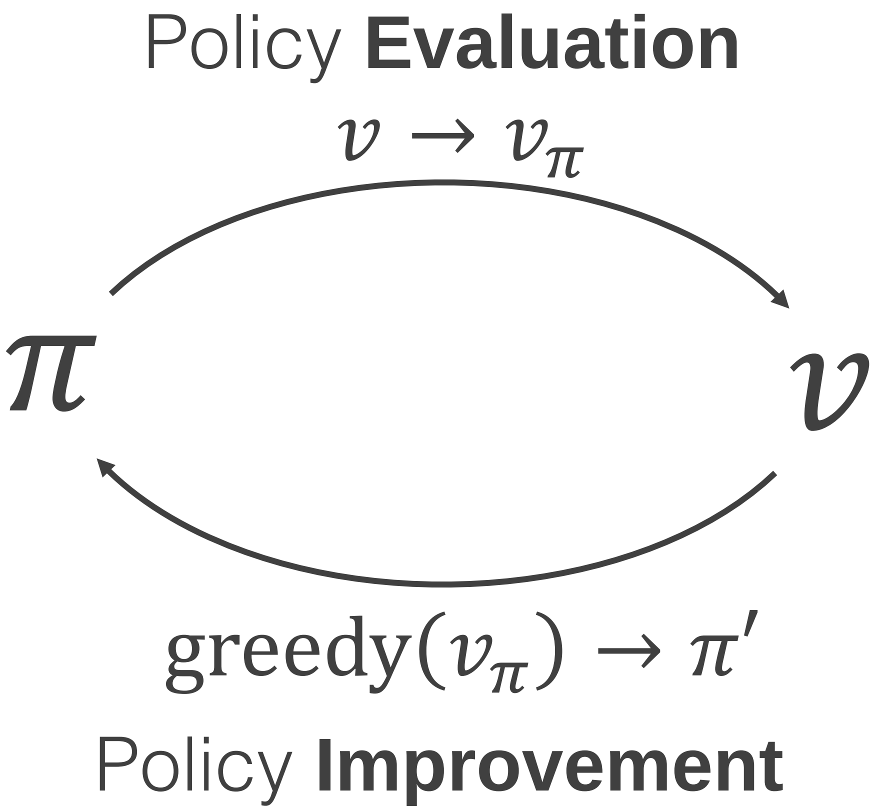
Adapted from David Silver, 2015 and Sutton and Barto, 1998

3. Policy Iteration

Find the **best** policy

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

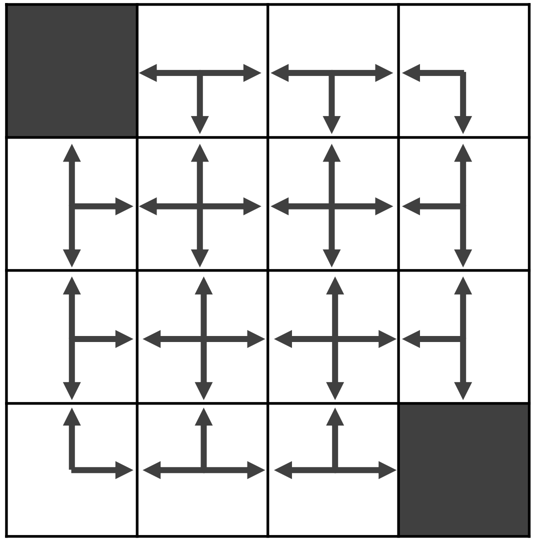
Output: **best** policy $\pi^*(a|s)$



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

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

1 $\pi_0(s)$



$v_0(s)$

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

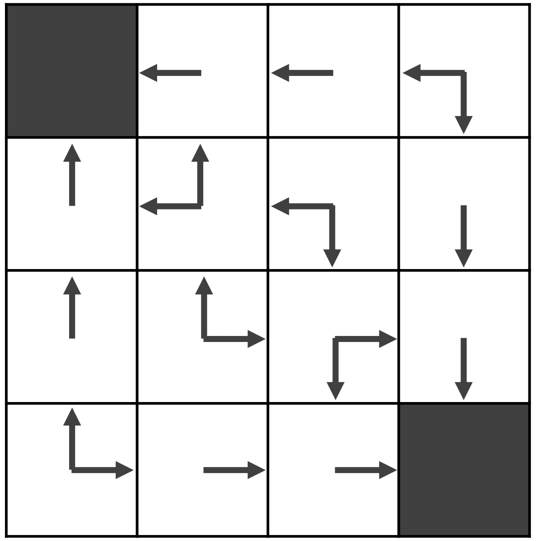
Policy Evaluation

2 $v_\infty(s) \rightarrow v_{\pi_0}(s)$

0	-14	-20	-22
-14	-18	-20	-20
-20	-20	-18	-14
-22	-20	-14	

Policy Improvement

3 $\pi_1(s) = \pi^*(s)$



$v_0(s)$

0	-14	-20	-22
-14	-18	-20	-20
-20	-20	-18	-14
-22	-20	-14	

Policy Evaluation

4 $v_\infty(s) \rightarrow v_{\pi_0}(s)$

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

0	-1	-2	-3
-1	-2	-3	-2
-2	-3	-2	-1
-3	-2	-1	0

Roadmap to optimal policies

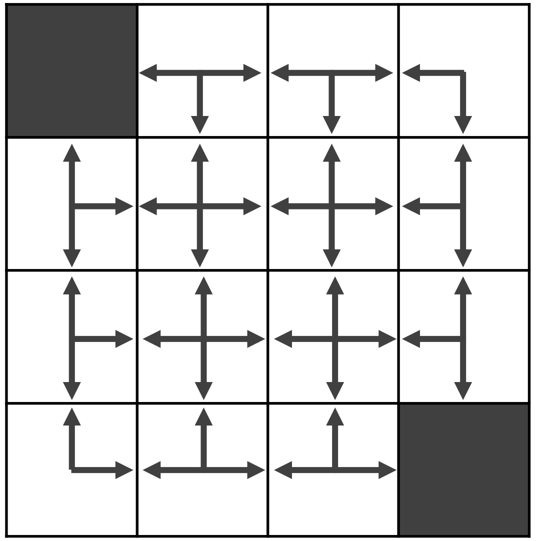
If we assume a **fully known MDP environment**, how do we…
(Markov Decision Process)

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**

$\pi_0(s)$



$v_0(s)$

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

$v_1(s)$

0	-1	-1	-1
-1	-1	-1	-1
-1	-1	-1	-1
-1	-1	-1	0

What if we stopped after one sweep. This is…

4. Value Iteration

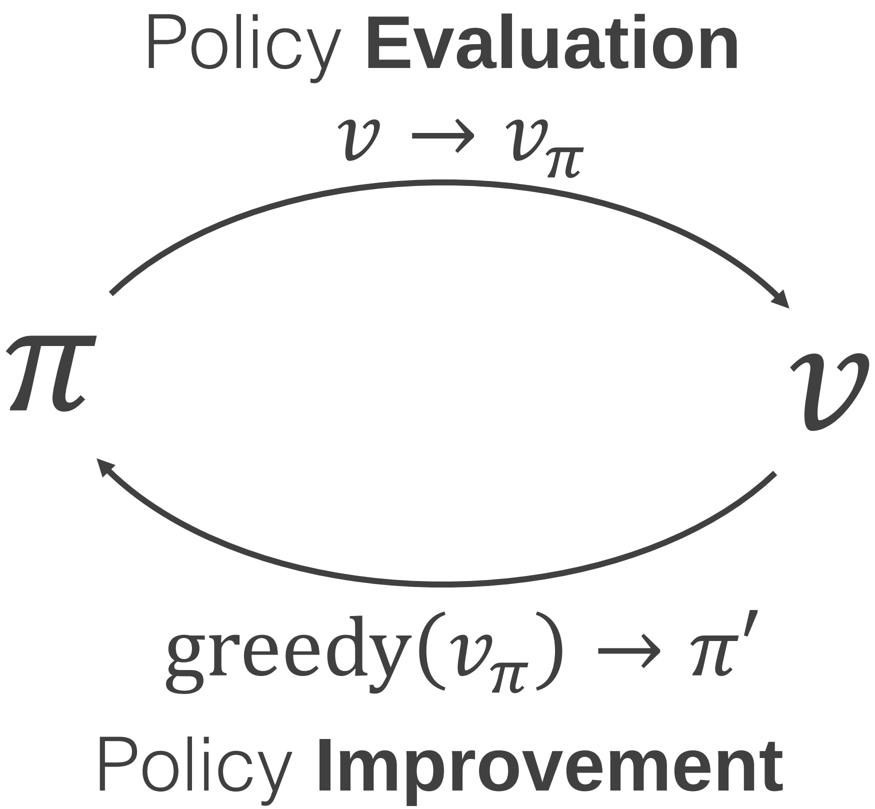
Find the best policy **faster**

4. Value Iteration

Find the best policy **faster**

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

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



1. **Policy Evaluation**: estimate v_π
   One-sweep of policy evaluation
2. **Policy Improvement**: generate $\pi' \geq \pi$
   Greedy policy improvement
3. Iterate 1 and 2 until convergence

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

$v_0(s)$

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

$v_1(s)$

0	-1	-1	-1
-1	-1	-1	-1
-1	-1	-1	-1
-1	-1	-1	0

$v_2(s)$

0	-1.7	-2	-2
-1.7	-2	-2	-2
-2	-2	-2	-1.7
-2	-2	-1.7	0

$v_3(s)$

0	-2.4	-2.9	-3.0
-2.4	-2.9	-3.0	-2.9
-2.9	-3.0	-2.9	-2.4
-3.0	-2.9	-2.4	0

$v_{10}(s)$

0	-6.1	-8.4	-9.0
-6.1	-7.7	-8.4	-8.4
-8.4	-8.4	-7.7	-6.1
-9.0	-8.4	-6.1	0

$v_\infty(s) = v_\pi(s)$

0	-14	-20	-22
-14	-18	-20	-20
-20	-20	-18	-14
-22	-20	-14	

So far, we've run policy evaluation all the way to convergence (**this is slow**)

1 $v_0(s)$

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

3 $v_1(s)$

0	-1	-1	-1
-1	-1	-1	-1
-1	-1	-1	-1
-1	-1	-1	0

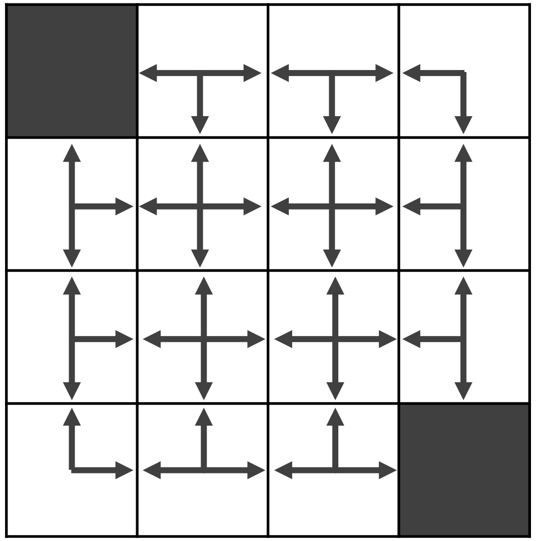
5 $v_2(s)$

0	-1	-2	-2
-1	-2	-2	-2
-2	-2	-2	-1
-2	-2	-1	0

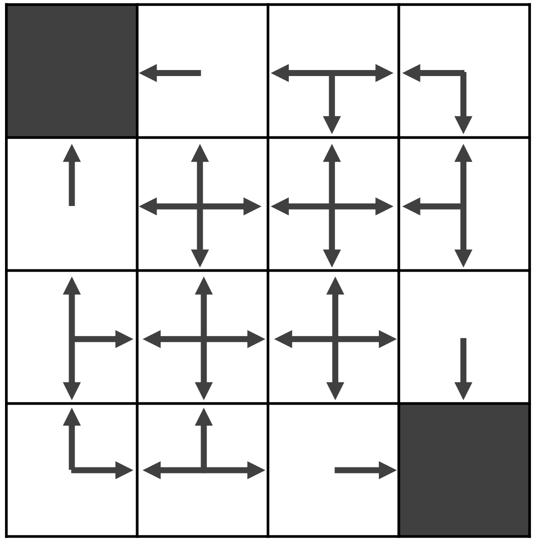
7 $v_3(s) = v_{\pi^*}(s)$

0	-1	-2	-3
-1	-2	-3	-2
-2	-3	-2	-1
-3	-2	-1	0

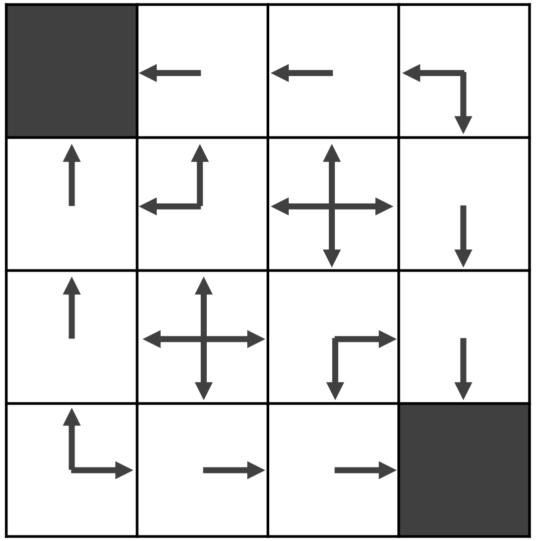
2 $\pi_0(s)$



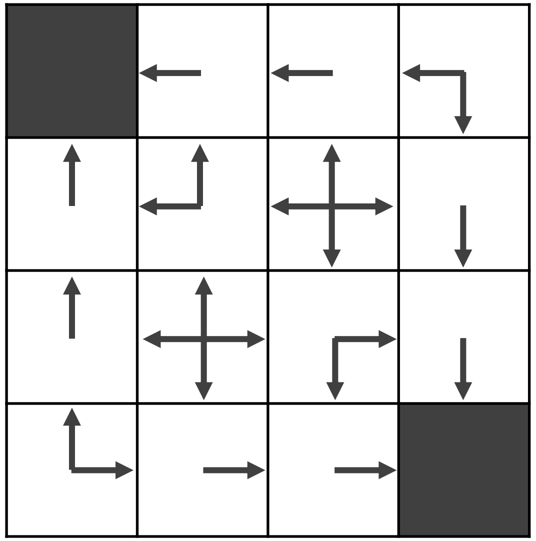
4 $\pi_1(s)$



6 $\pi_2(s)$



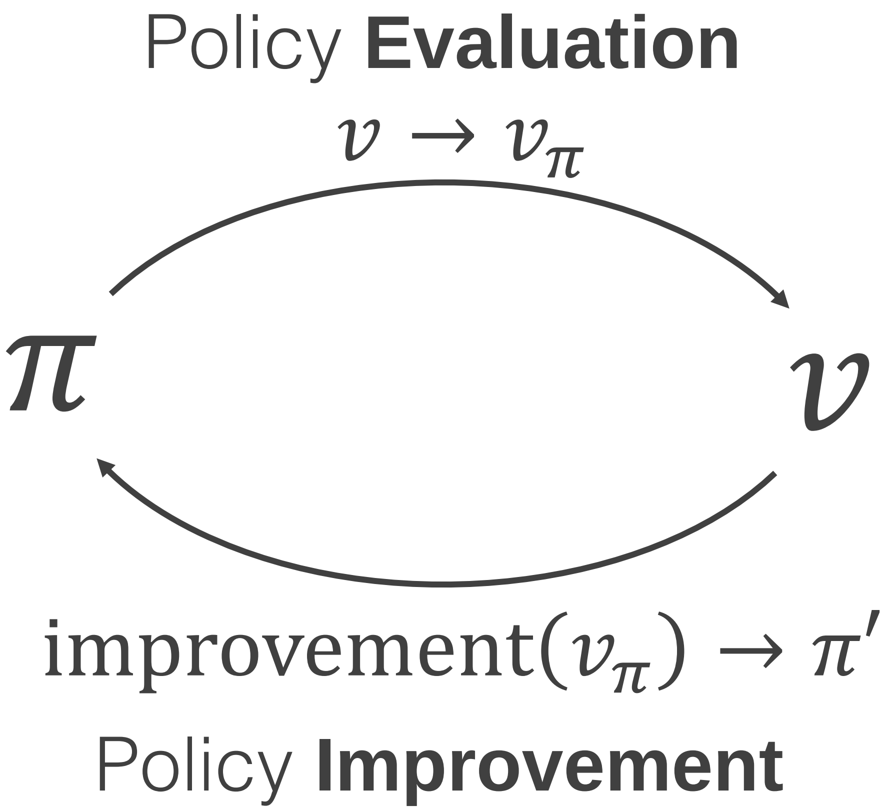
8 $\pi_3(s) = \pi^*(s)$



Generalized Policy Iteration

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

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



1. **Policy Evaluation**: estimate v_π
   Any policy evaluation algorithm
2. **Policy Improvement**: generate $\pi' \geq \pi$
   Any policy improvement algorithm
3. Iterate 1 and 2 until convergence

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

Demo

https://cs.stanford.edu/people/karpathy/reinforcejs/gridworld_dp.html

So far, we've assumed full knowledge of the environment (MDP)

What if we **DO NOT assume full knowledge of the environment** (MDP)

This means we have to **learn by 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

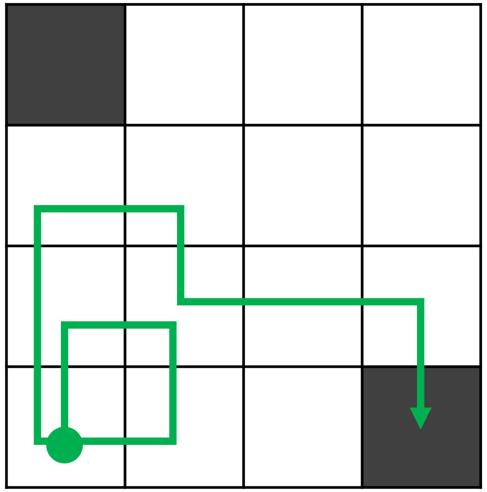
Monte Carlo Policy Evaluation

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



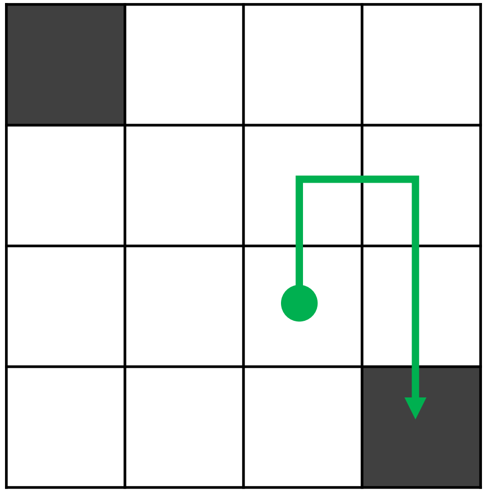
Episode 1 **returns** after the first visit of each state

$G^{(1)}$

-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, $G^{(1)}$

$v_2(s)$

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

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

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**

$$\pi'(s) = \underset{a}{\operatorname{argmax}}\, \underset{?}{R_s^a} + \underset{?}{P_{ss'}^a} v(s')$$

Greedy policy improvement over $q(s,a)$ **requires no MDP knowledge**

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

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 $\mathrm{AVG}(\mathrm{Returns}(\mathrm{s}, \mathrm{a})) \rightarrow \hat{q}_\pi(s,a)$

Sutton and Barto, 1998

3. Policy Iteration
Find the **best** policy

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

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

Policy **Evaluation**

$v \rightarrow v_\pi$

π v

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

Policy **Improvement**

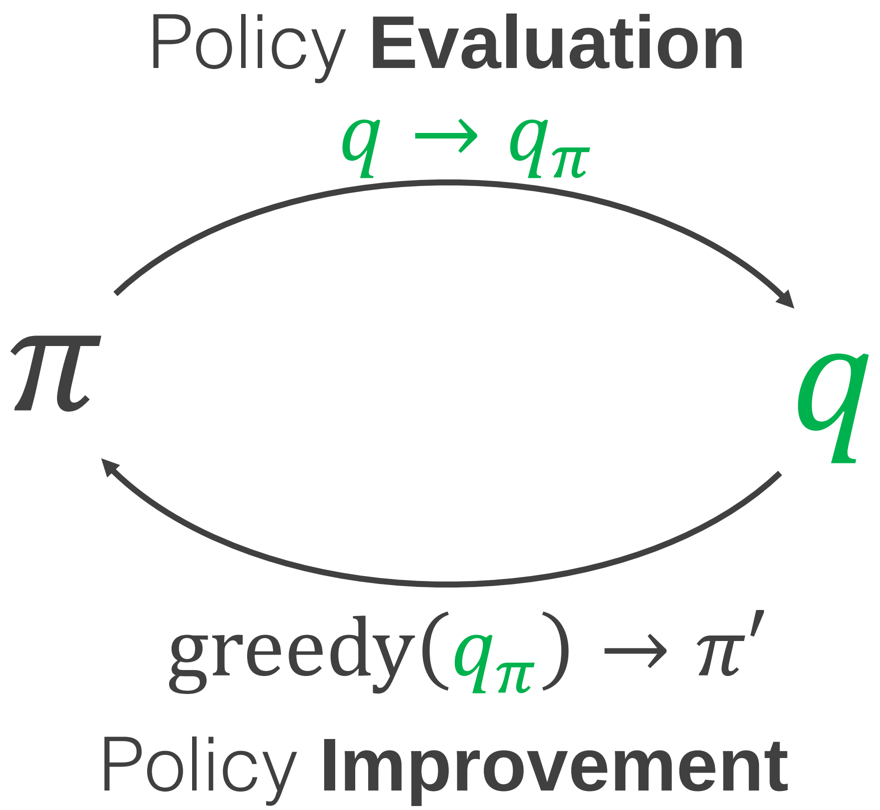
1. **Policy Evaluation**: estimate v_π
   Iterative policy evaluation
   Note: This is VERY slow
2. **Policy Improvement**: generate $\pi' \geq \pi$
   Greedy policy improvement
3. Iterate 1 and 2 until convergence

PREVIOUSLY

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

Monte Carlo Control

Find the **best** policy

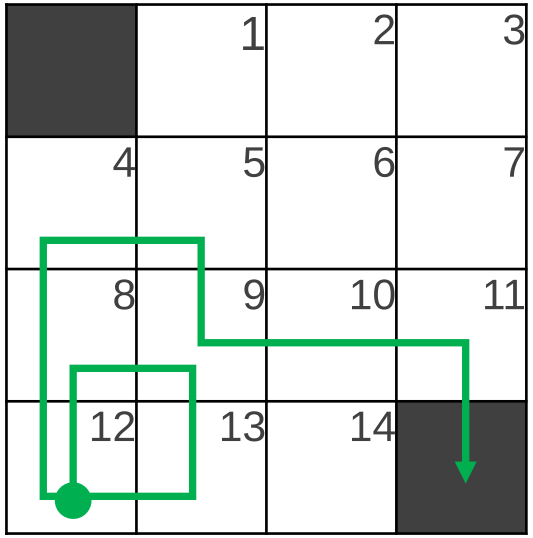


1. **Policy Evaluation**: estimate q_π
   Monte Carlo action policy evaluation
2. **Policy Improvement**: generate $\pi' \geq \pi$
   Greedy policy improvement
3. Iterate 1 and 2 until convergence

Sutton and Barto, 1998

Monte Carlo Control

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



State labels

Episode 1

Total Reward: -11

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): ↑ → ← ↓

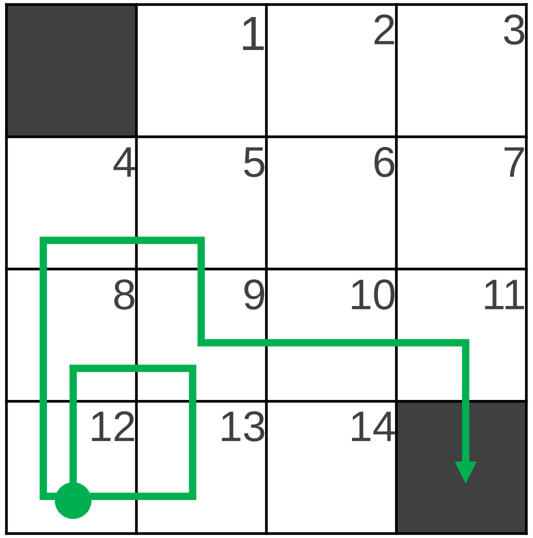
State (s)	↑	→	←	↓
1				
2				
3				
4				
5				
6				
7				
8				
9				
10				
11				
12				
13				
14				

Monte Carlo Control

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

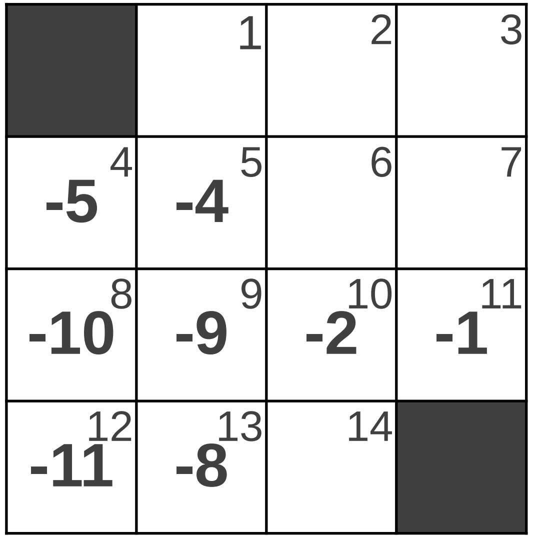
Episode 1

Total Reward: -11



State labels

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



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 $q(s,a)$

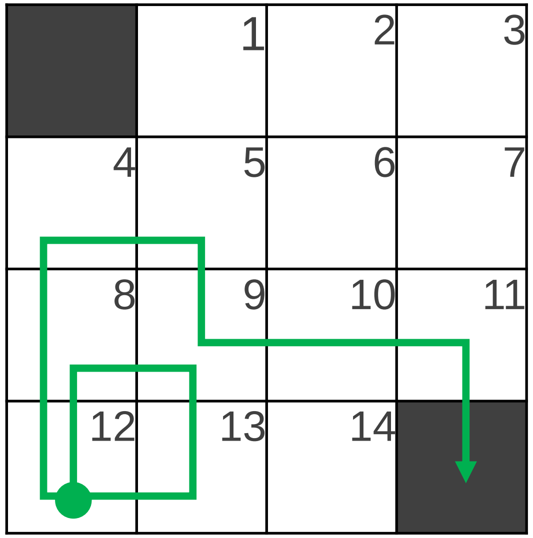
$q_\pi(s,a)$

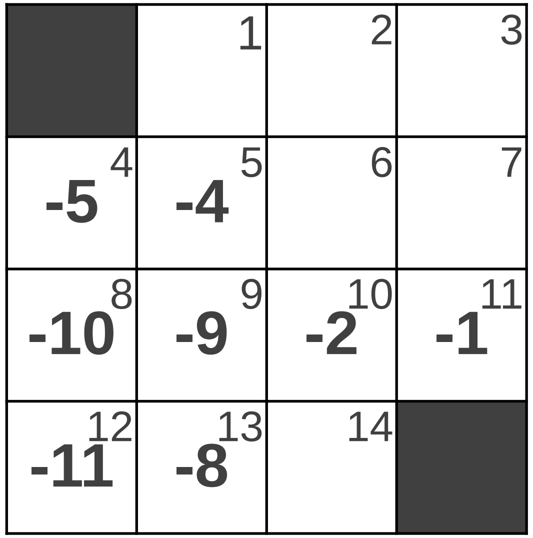
Invalid action

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





Returns from episode 1

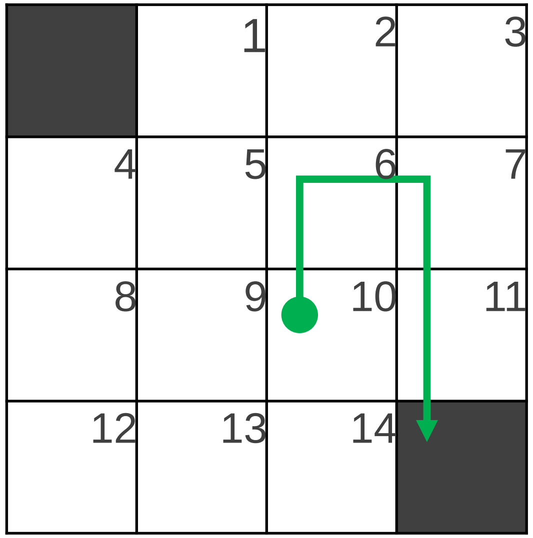
Action (a): ↑ → ← ↓

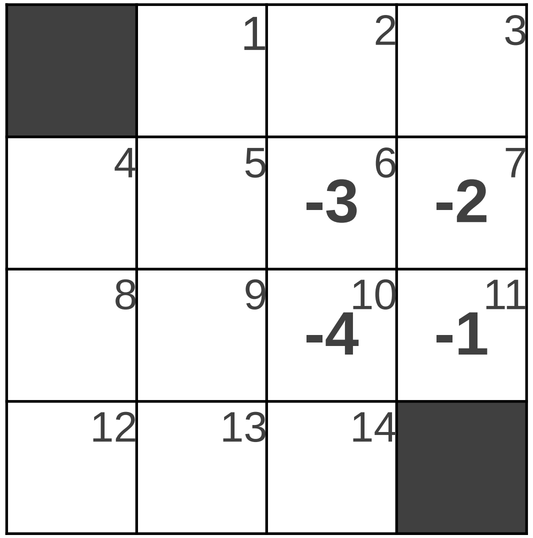
$q_\pi(s,a)$

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 2





Returns from episode 2

Action (a): ↑ → ← ↓

$q_\pi(s,a)$

State (s)

	↑	→	←	↓
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				

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

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

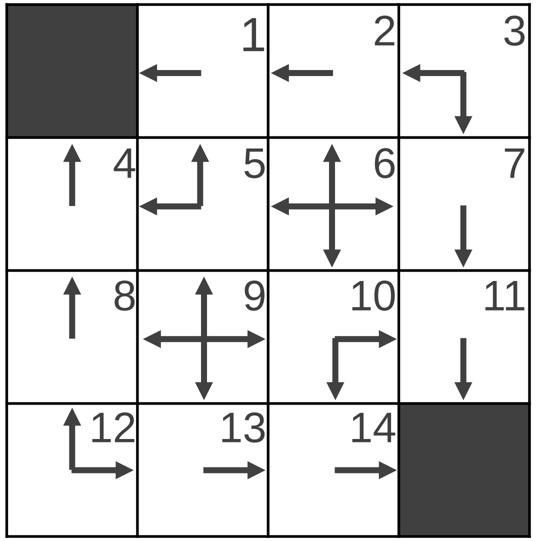
State (s) \ Action (a)	↑	→	←	↓
1		-3	-1	-3
2		-4	-2	-3
3			-3	-3
4	-1	-3		-3
5	-2	-4	-2	-4
6	-3	-3	-3	-3
7	-4		-4	-2
8	-2	-4		-4
9	-3	-3	-3	-3
10	-4	-2	-4	-2
11	-3		-3	-1
12	-3	-3		
13	-4	-2	-4	
14	-3	-1	-3	

If we 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 (e.g. Q-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

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**	Dynamic Programming
2. Find a **better** policy?	**Policy improvement**	
3. Find the **best** policy?	**Policy iteration**	
4. Find the best policy **faster**?	**Value iteration**	

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

Reinforcement Learning Roadmap

Knowledge of **Environment**

Perfect knowledge
Known Markov Decision Process

↓

No knowledge
Must learn from experience

Dynamic Programming

What's a Markov Decision Process?
How do we find optimal policies?

Monte Carlo Control

How do we estimate our value functions?
How do we use the value functions to choose actions?