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

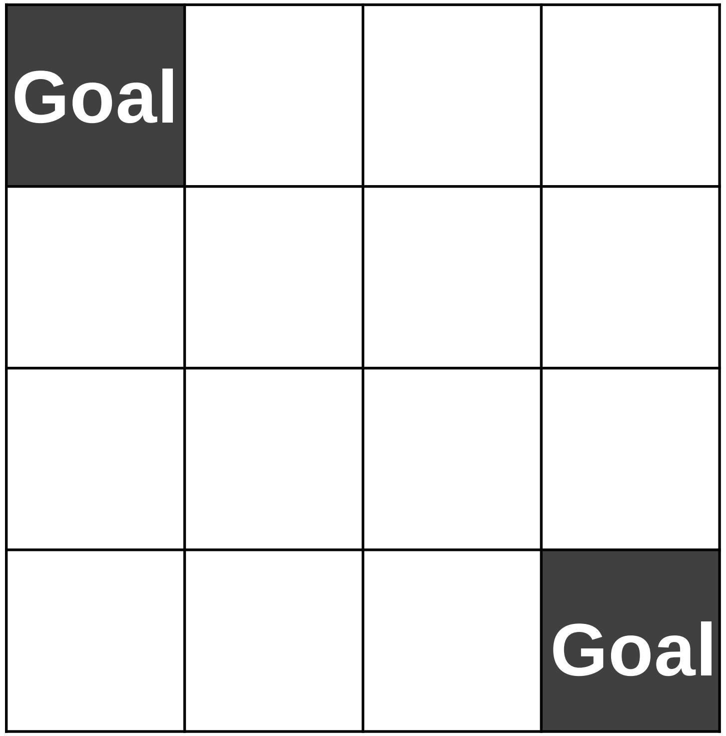
How do we find optimal policies?
(Bellman equations)

No knowledge
Must learn from experience

4 Monte Carlo Control

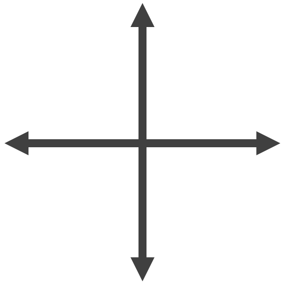
How do we estimate our value functions?
How do we use the value functions to choose actions?
How do we learn optimal policies from experience?

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

$v_0(s)$

(initialization)

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

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

for any action a

(i.e. randomly go in any valid direction)

Value function initialization:

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

$v_k(s) \rightarrow$ iteration k of policy evaluation

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

1. Policy Evaluation

Evaluate the returns a policy will yield

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

(randomly go in any direction)

$v_0(s)$

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

Bellman Expectation Equation:

$$v_{k+1}(s) = \sum_a \pi(a|s) \sum_{s'} \sum_r p(s', r|s, a)[r + \gamma v_k(s')]$$

1 $(\gamma = 1)$

-1 (rewards are deterministic and constant for all actions)

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

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

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

Average of the value of the N_a neighboring states

Here, the next state is a deterministic function of a, so we can think of it as $s'(a)$

1. Policy Evaluation

Evaluate the returns a policy will yield

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

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

$v_0(s)$

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 yields the **highest expected returns**

Adapted from David Silver, 2015

2. Policy Improvement

Find a **better** policy

Value function: $v_0(s)$

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

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

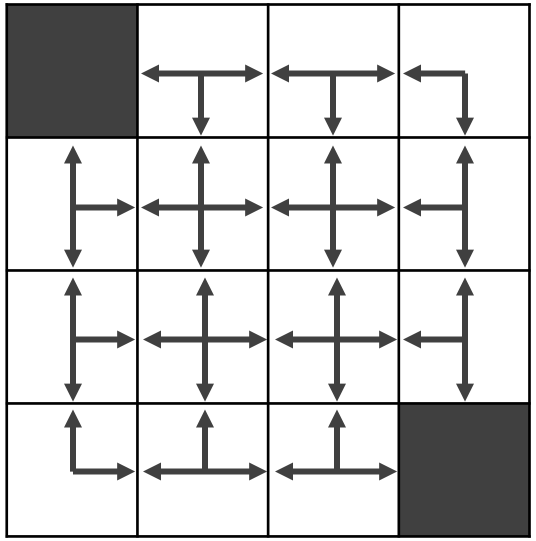
0	s_1 -14	-20	-22
-14	-18	-20	-20
-20	-20	-18	-14
-22	-20	-14	0

Here, $q_\pi(s,a) = -1 + v_\pi(s')$ since each action leads deterministically to one state, s'

$$q_\pi(s_1, a) = \begin{cases} -1 & \leftarrow \\ -19 & \downarrow \\ -21 & \rightarrow \end{cases}$$

Initial policy: $\pi(s)$

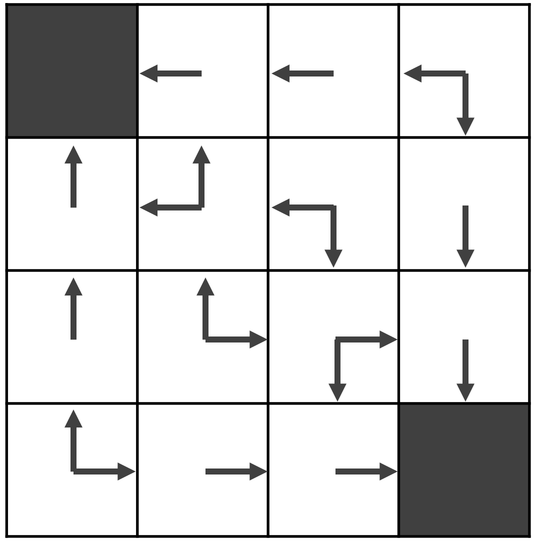
$\pi(a|s) =$ randomly go in any valid direction



Improved policy

(in this case this is an optimal policy)

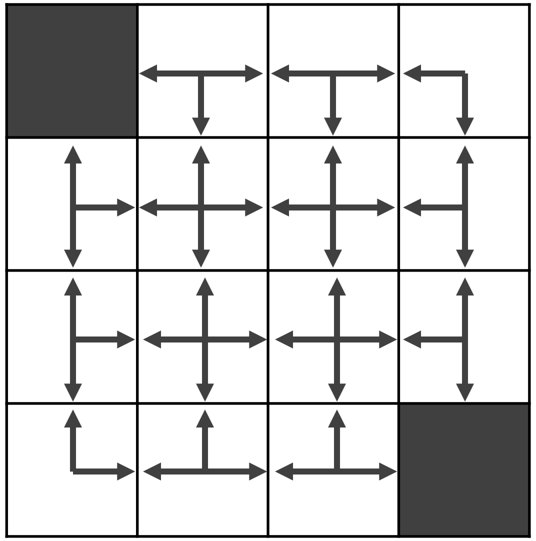
$\pi'(s)$



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

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

$$\pi(a|s)$$



State Labels (s)

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

$\pi(a|s)$

randomly go in any valid direction

Invalid action

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

$q_\pi(s,a)$

State (s) \ Action (a):	↑	→	←	↓
1		-21	-1	-19
2		-23	-15	-21
3			-21	-21
4	-1	-19		-21
5	-15	-21	-15	-21
6	-21	-21	-19	-19
7	-23		-21	-15
8	-15	-21		-23
9	-19	-19	-21	-21
10	-21	-15	-21	-15
11	-21		-19	-1
12	-21	-21		
13	-21	-15	-23	
14	-19	-1	-21	

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**	

3. Policy Iteration

Find the **best** policy

Policy **Evaluation**

$v \rightarrow v_\pi$

π $\qquad$ v

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

Policy **Improvement**

⋮

This process will converge to the optimal functions

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

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**

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

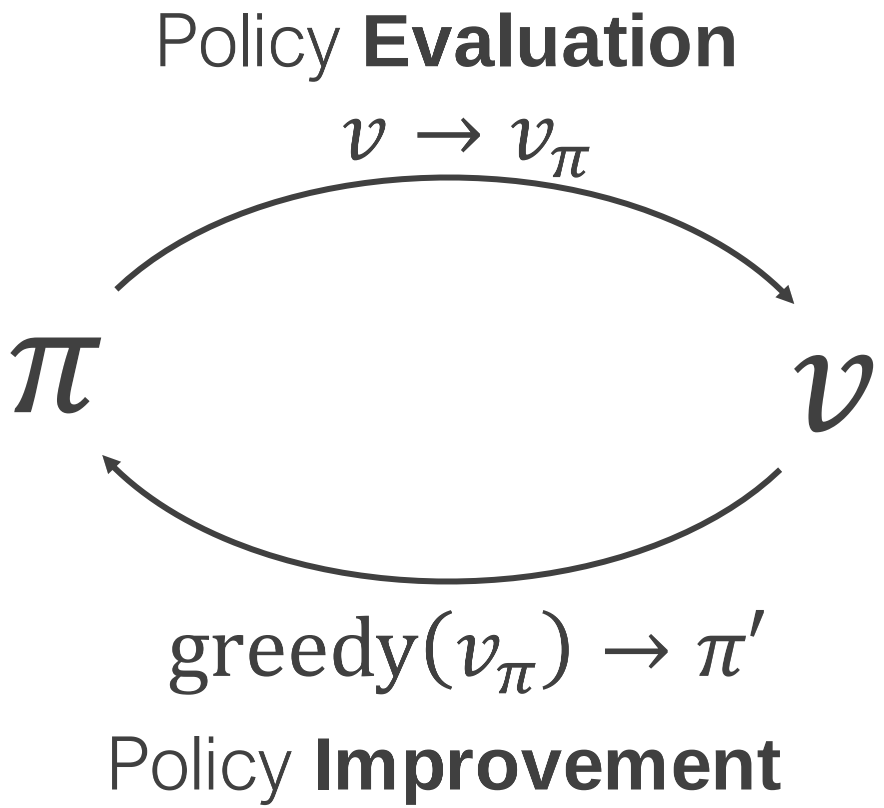
Adapted from David Silver, 2015 and Sutton and Barto, 1998

3. Policy Iteration

Find the **best** policy

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

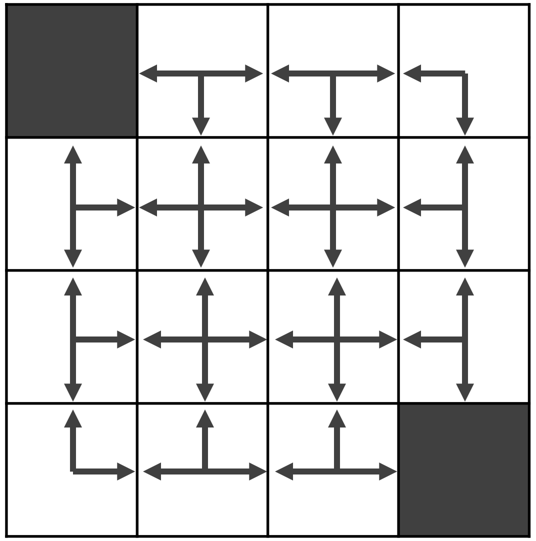
Output: **best** policy $\pi^*(a|s)$



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 $\pi_0(s)$



Policy Evaluation

$v_0(s)$

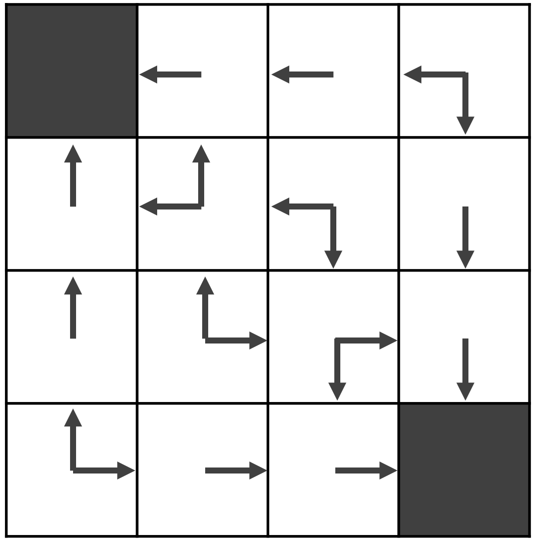
0	0	0	0
0	0	0	0
0	0	0	0
0	0	0	0

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 $v_\infty(s) \rightarrow v_{\pi_1}(s)$

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

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

Roadmap to optimal policies

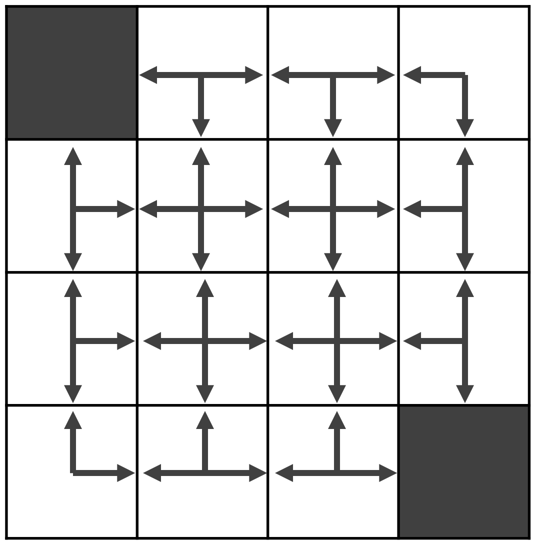
If we assume a **fully known MDP environment**, how do we…
(Markov Decision Process)

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

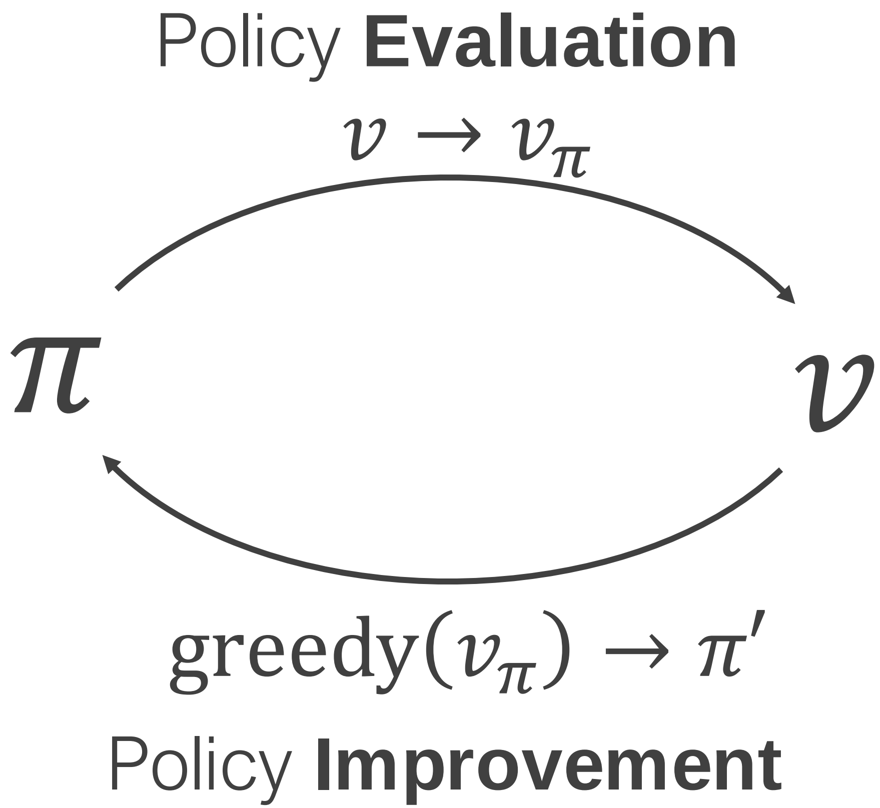
Find the best policy **faster**

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

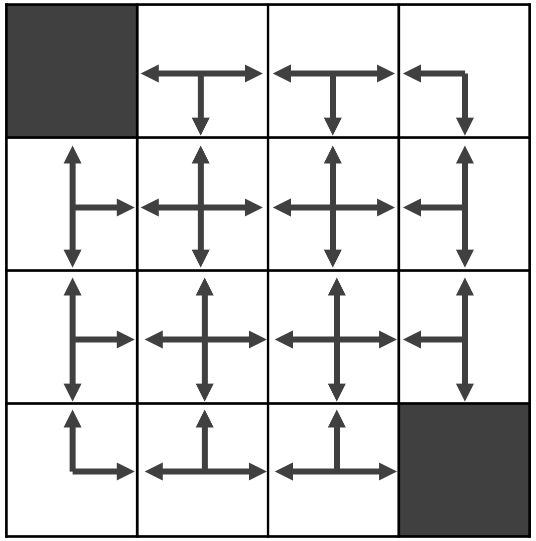
5 $v_2(s)$

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

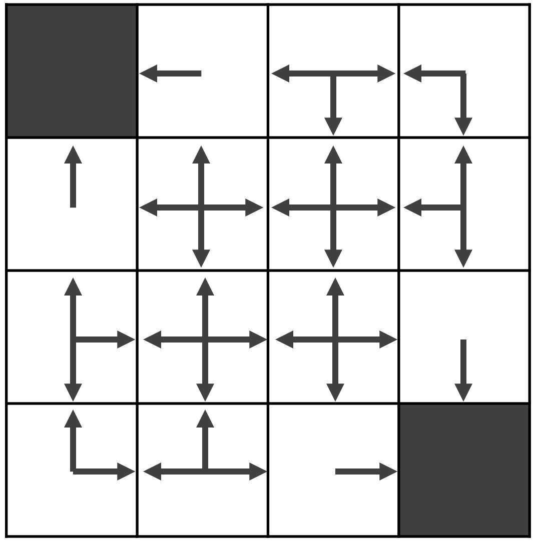
7 $v_3(s) = v_{\pi^*}(s)$

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

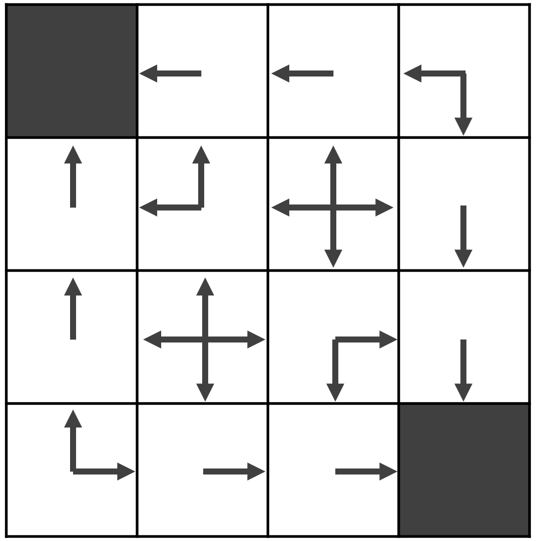
2 $\pi_0(s)$



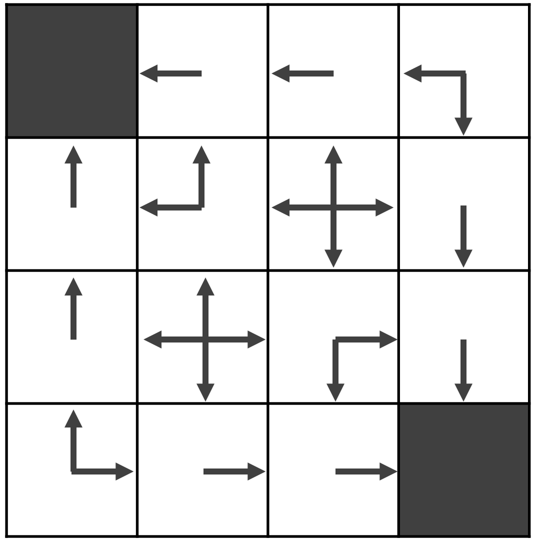
4 $\pi_1(s)$



6 $\pi_2(s)$



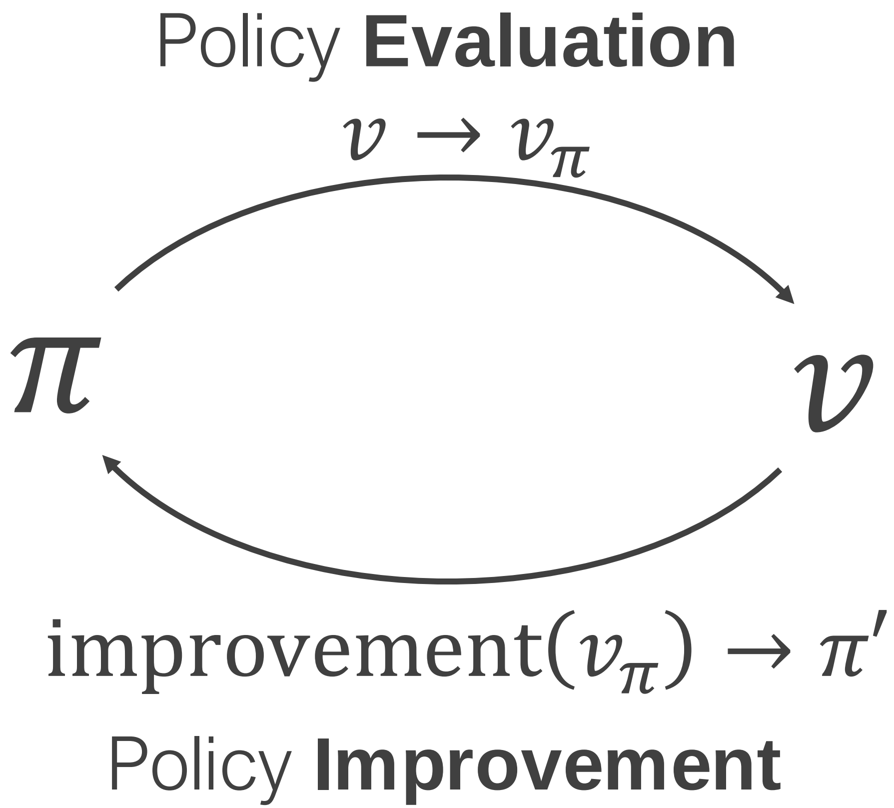
8 $\pi_3(s) = \pi^*(s)$



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 $\mathrm{AVG}(\mathrm{Returns}(s)) \rightarrow \hat{v}_\pi(s)$

Sutton and Barto, 1998

Monte Carlo Policy Evaluation

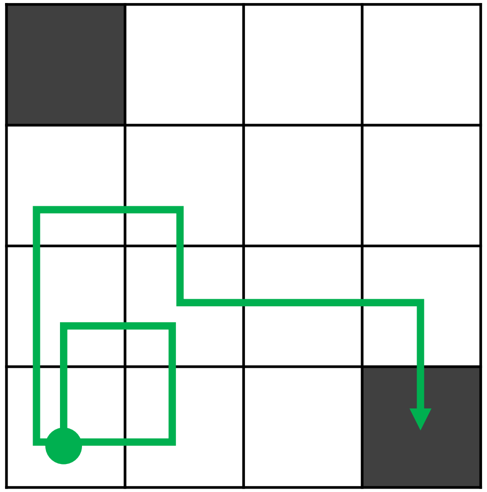
For **state** values

"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

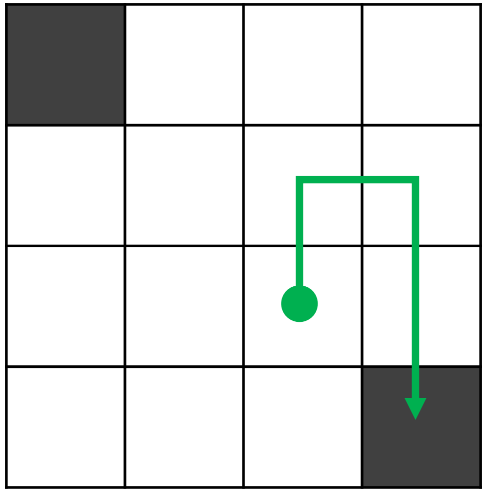
$G^{(1)}$

-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, $G^{(1)}$

$v_2(s)$

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

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

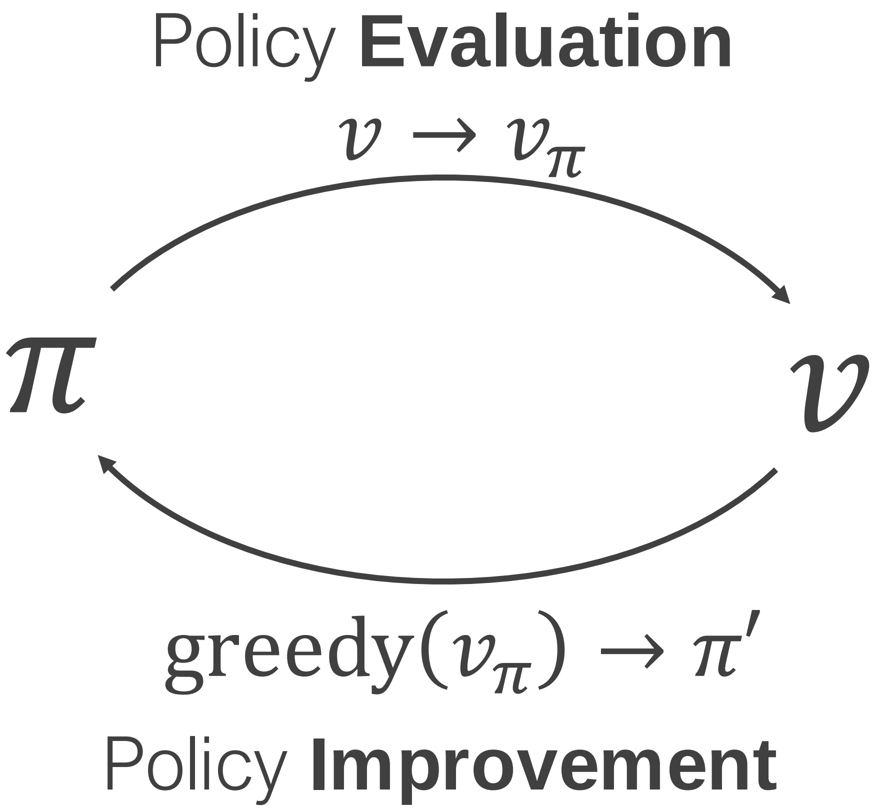
Sutton and Barto, 1998

3. Policy Iteration

Find the **best** policy

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

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



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 q

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

Policy **Improvement**

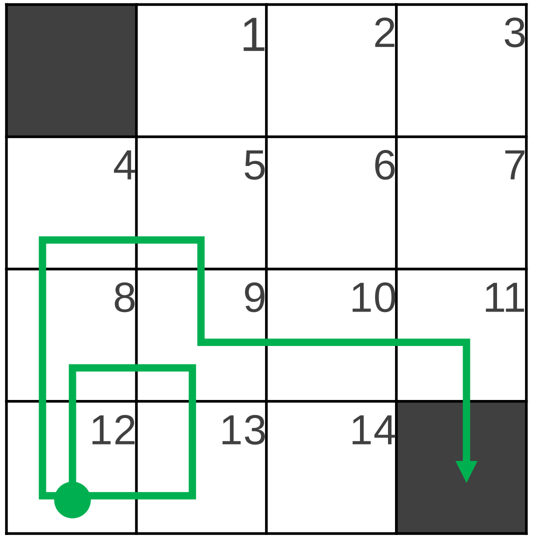
1. **Policy Evaluation**: estimate q_π
   Monte Carlo action policy evaluation
2. **Policy Improvement**: generate $\pi' \geq \pi$
   Greedy policy improvement
3. Iterate 1 and 2 until convergence

Policy needs to be ε-greedy to converge on the optimal policy

Sutton and Barto, 1998

Monte Carlo Control

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



← State labels

Episode 1
Total Reward: -11

❶ MC Policy Evaluation

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

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

$q_\pi(s,a)$

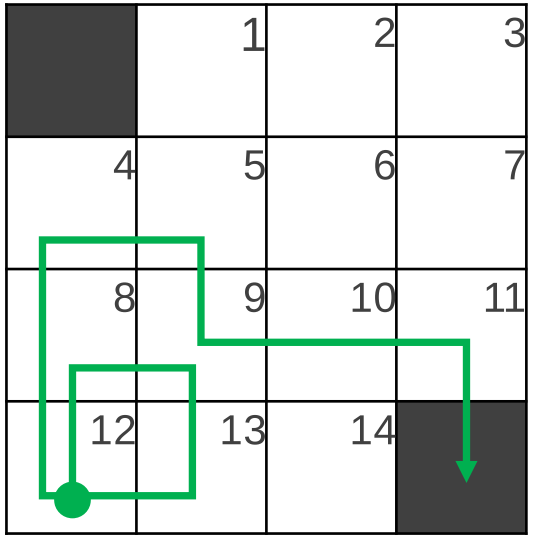
Action (a): ↑ → ← ↓

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

Discount rate: $\gamma = 1$

Monte Carlo Control

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



State labels

Episode 1

Total Reward: -11

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 $q_\pi(s, a)$

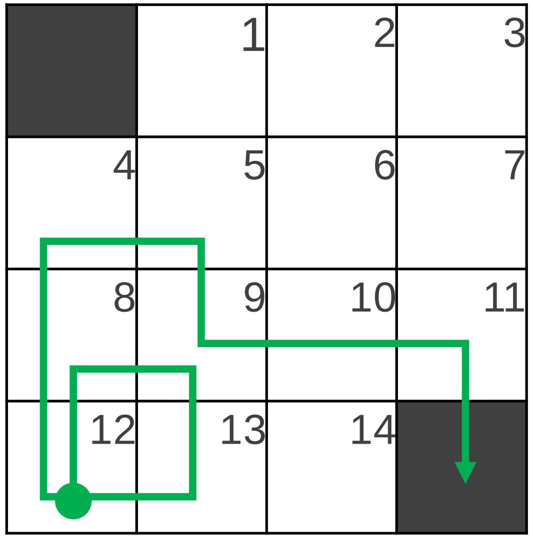
$q_\pi(s, a)$

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	

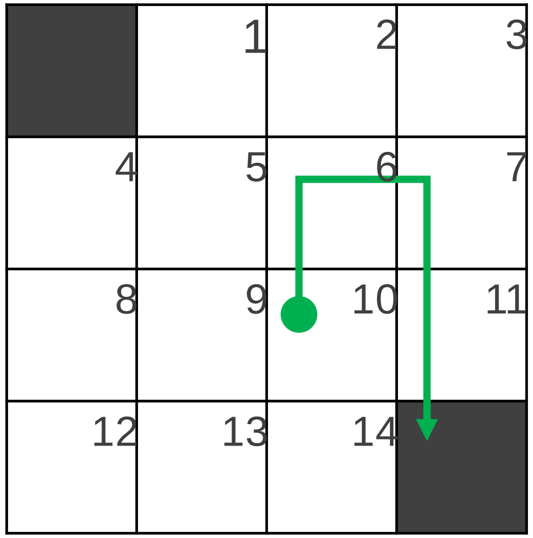
Returns from episode 1

Action (a): ↑ → ← ↓

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

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



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

Returns from episode 2

Action (a): ↑ → ← ↓

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

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

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

Action (a): ↑ → ← ↓

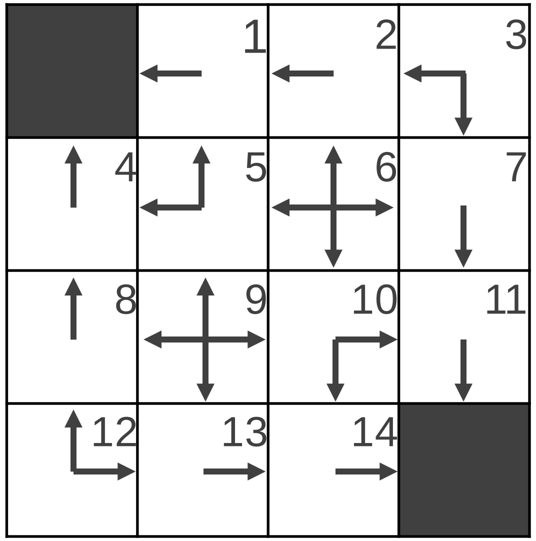
State (s)

State (s)	↑	→	←	↓
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	

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

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?