Reinforcement Learning III

Reinforcement Learning Roadmap

Knowledge of **Environment**

Perfect knowledge
Known Markov
Decision Process

No knowledge
Must learn from
experience

1 Core concepts in reinforcement learning

Actions, Rewards, Value, Environments, and Policies

2 Markov decision processes

…and Markov chains and Markov reward processes

3 Dynamic Programming

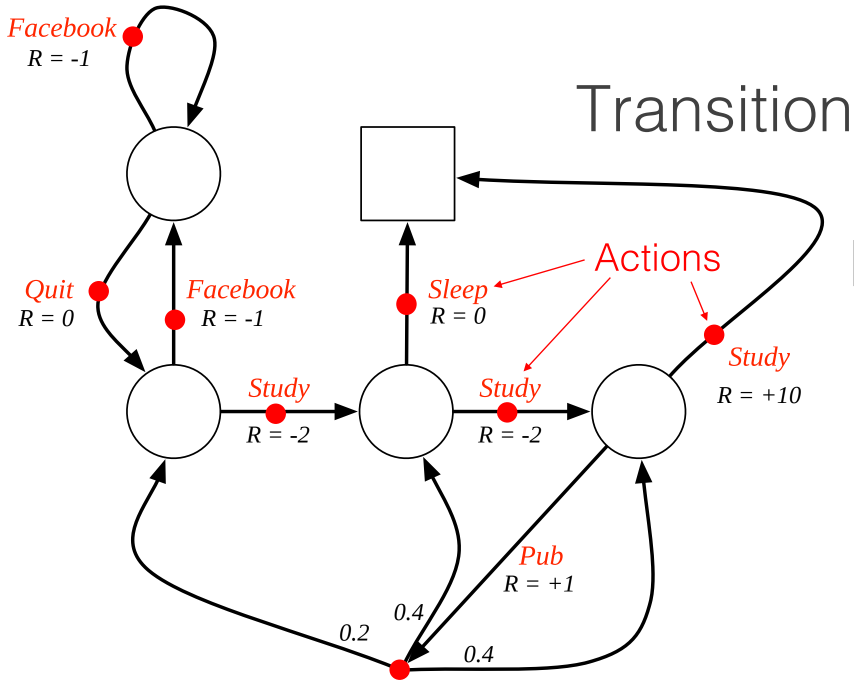
How do we find optimal policies?
(Bellman equations)

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?

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)

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

$$v_\pi(s) = E[R_{t+1} + \gamma v_\pi(S_{t+1})|s]$$

Action value function

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

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

$$q_\pi(s,a) = E[R_{t+1} + \gamma q_\pi(S_{t+1}, A_{t+1})|s,a]$$

David Silver, UCL, 2015

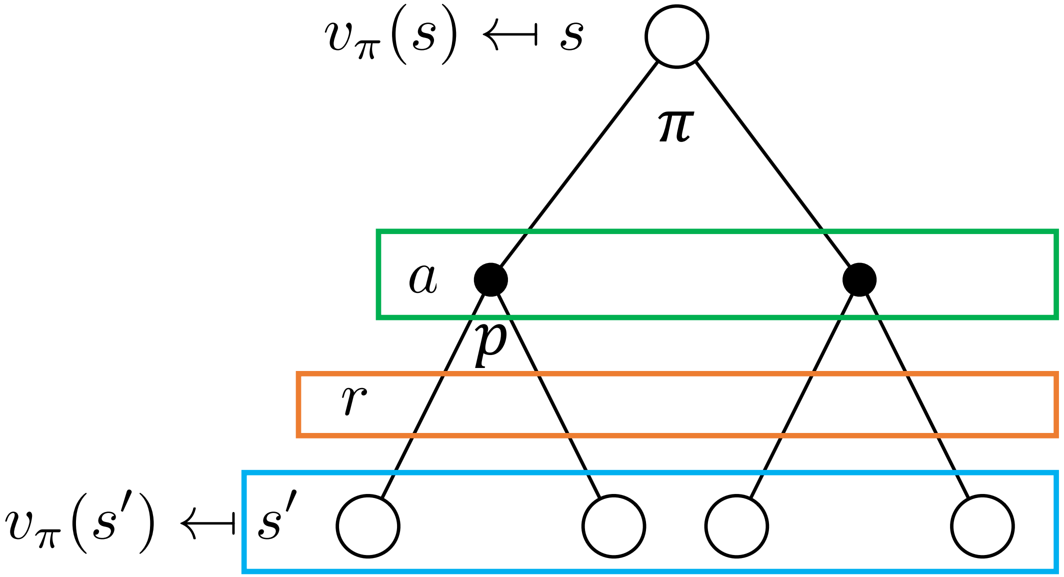
Bellman Expectation Equations for the state value function

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

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

$$v_\pi(s) = E_\pi[R_{t+1} + \gamma G_{t+1} | S_t = s]$$

$$= E_\pi[R_{t+1} + \gamma v_\pi(S_{t+1}) | S_t = s]$$



Expectation over the possible actions

Expectation over the rewards

(based on state and choice of action)

Expectation over the next possible states

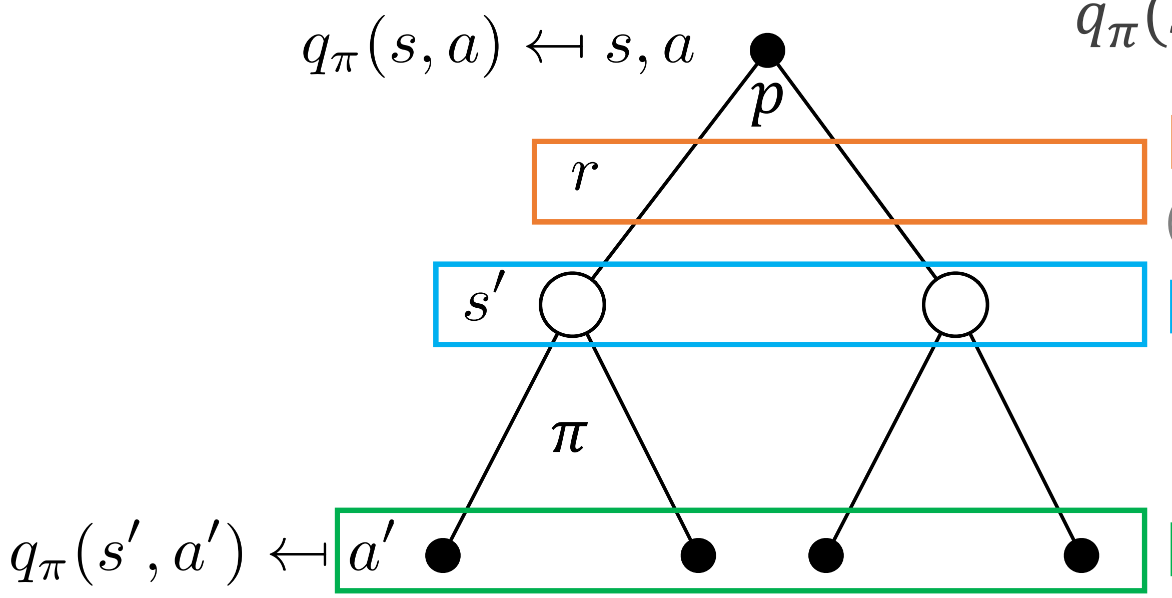
$$v_\pi(s) = \underbrace{\sum_a}\pi(a|s) \underbrace{\sum_{s'}} \underbrace{\sum_r} p(s', r | s, a)[r + \gamma v_\pi(s')]$$

Bellman Expectation Equations for the action value function

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

$$q_\pi(s,a) = E_\pi[G_t | S_t = s, A_t = a]$$

$$q_\pi(s,a) = E_\pi[R_{t+1} + \gamma q_\pi(S_{t+1}, A_{t+1}) | S_t = s, A_t = a]$$



Expectation over the rewards

(based on state and choice of action)

Expectation over the next possible states

Expectation over the possible actions

$$q_\pi(s,a) = \sum_{s'} \sum_{r} p(s', r | s, a) \left[r + \gamma \sum_{a'} \pi(a'|s') q_\pi(s',a') \right]$$

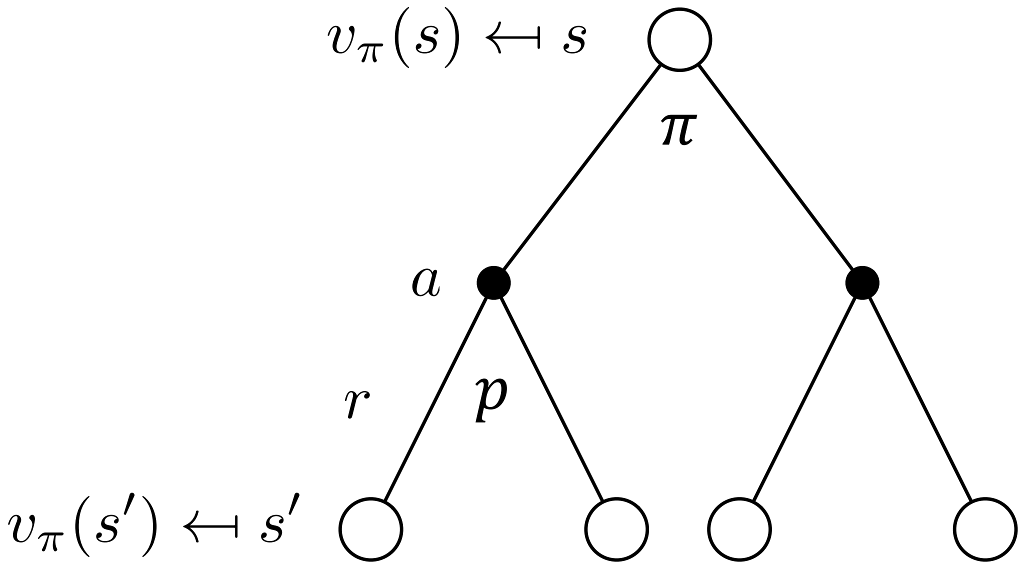
Bellman Expectation Equations

State value function

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

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

$$v_\pi(s) = E_\pi[R_{t+1} + \gamma v_\pi(S_{t+1}) | S_t = s]$$



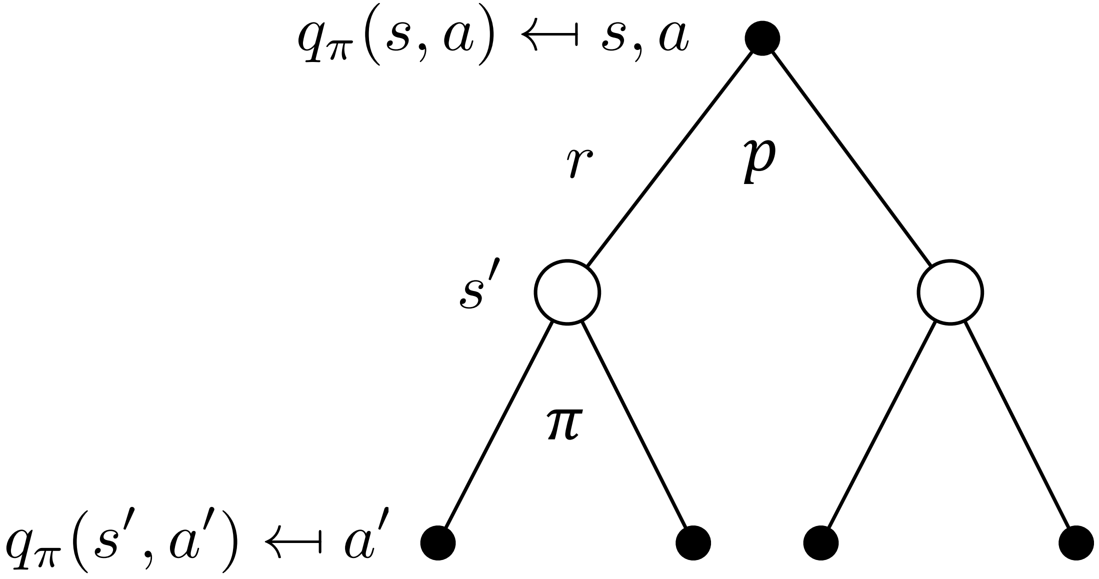
$$v_\pi(s) = \sum_a \pi(a|s) \underbrace{\sum_{s'} \sum_r p(s', r | s, a)[r + \gamma v_\pi(s')]}_{q_\pi(s,a)}$$

Action value function

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

$$q_\pi(s,a) = E_\pi[G_t | S_t = s, A_t = a]$$

$$q_\pi(s,a) = E_\pi[R_{t+1} + \gamma q_\pi(S_{t+1}, A_{t+1}) | S_t = s, A_t = a]$$

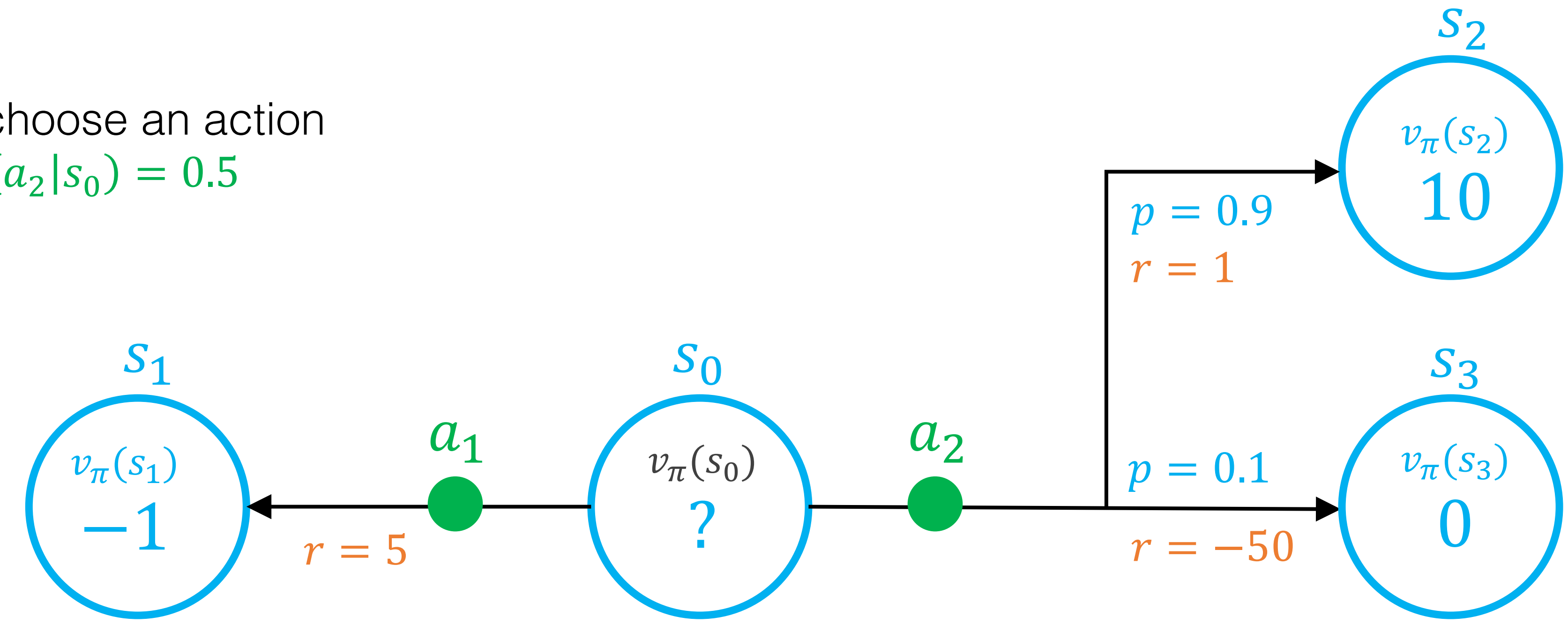


$$q_\pi(s,a) = \sum_{s'} \sum_r p(s', r | s, a) \left[r + \gamma \underbrace{\sum_{a'} \pi(a'|s') q_\pi(s', a')}_{v_\pi(s')} \right]$$

Example

Policy: randomly choose an action

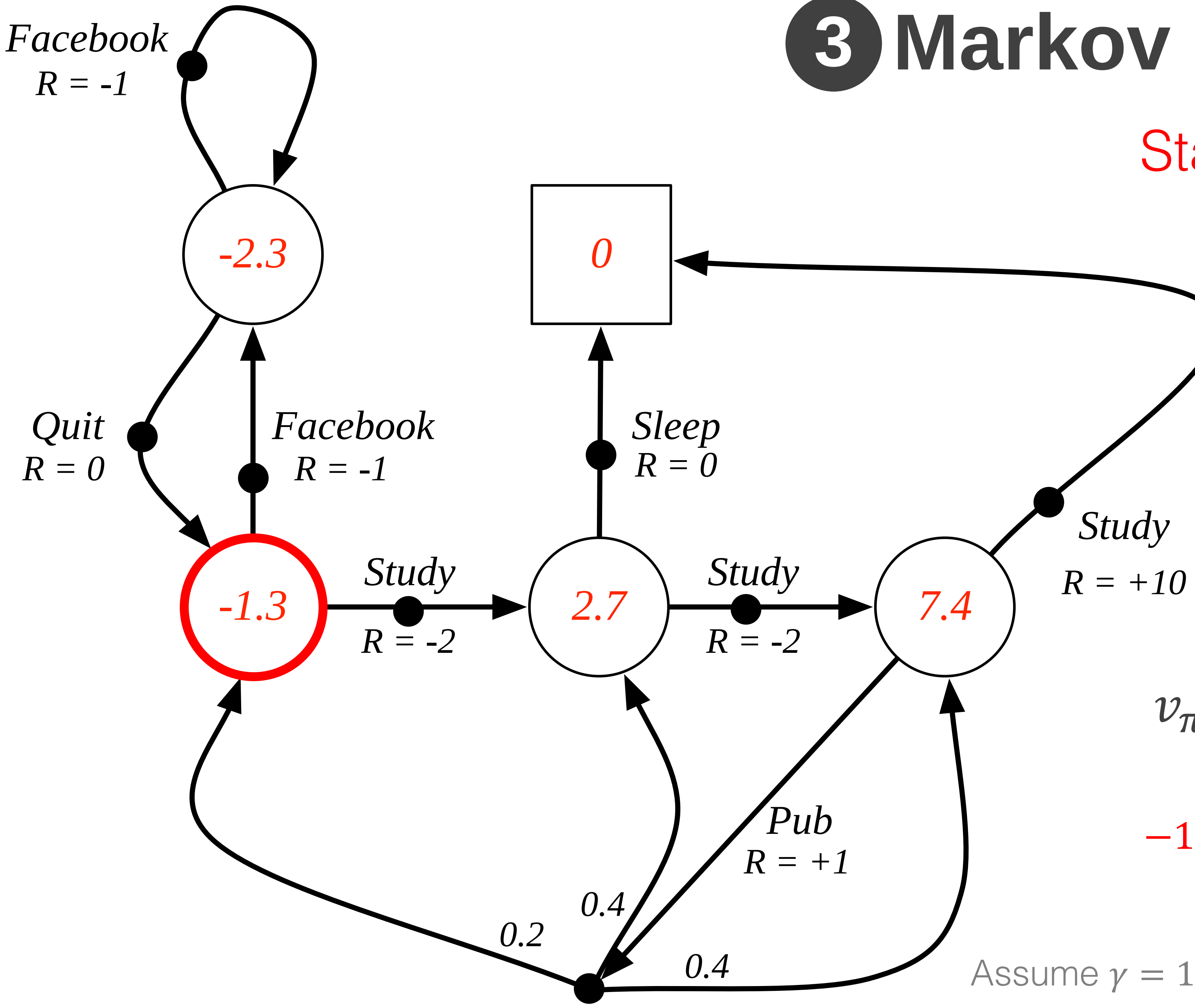
$\pi(a_1|s_0) = \pi(a_2|s_0) = 0.5$



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

$$v_\pi(s_0) = \overset{\pi(a_1|s_0)}{(0.5)}\underbrace{(\underset{r}{5} - \underset{v_\pi(s_1)}{1})}_{q_\pi(s_0,a_1)} + \overset{\pi(a_2|s_0)}{(0.5)}\underbrace{[(0.9)(1) + (0.1)(-50) + (\underset{p}{0.9})(\underset{v_\pi(s_2)}{10}) + (\underset{p}{0.1})(\underset{v_\pi(s_3)}{0})]}_{q_\pi(s_0,a_2)} \qquad \gamma = 1$$

3 Markov Decision Process



State value function, $v_\pi(s)$

Assumptions:

$$\pi(a|s) = \frac{1}{\text{\# of reachable states}}$$

(randomly select an action)

$$\gamma = 1$$

(no discounting)

Bellman Expectation Equation

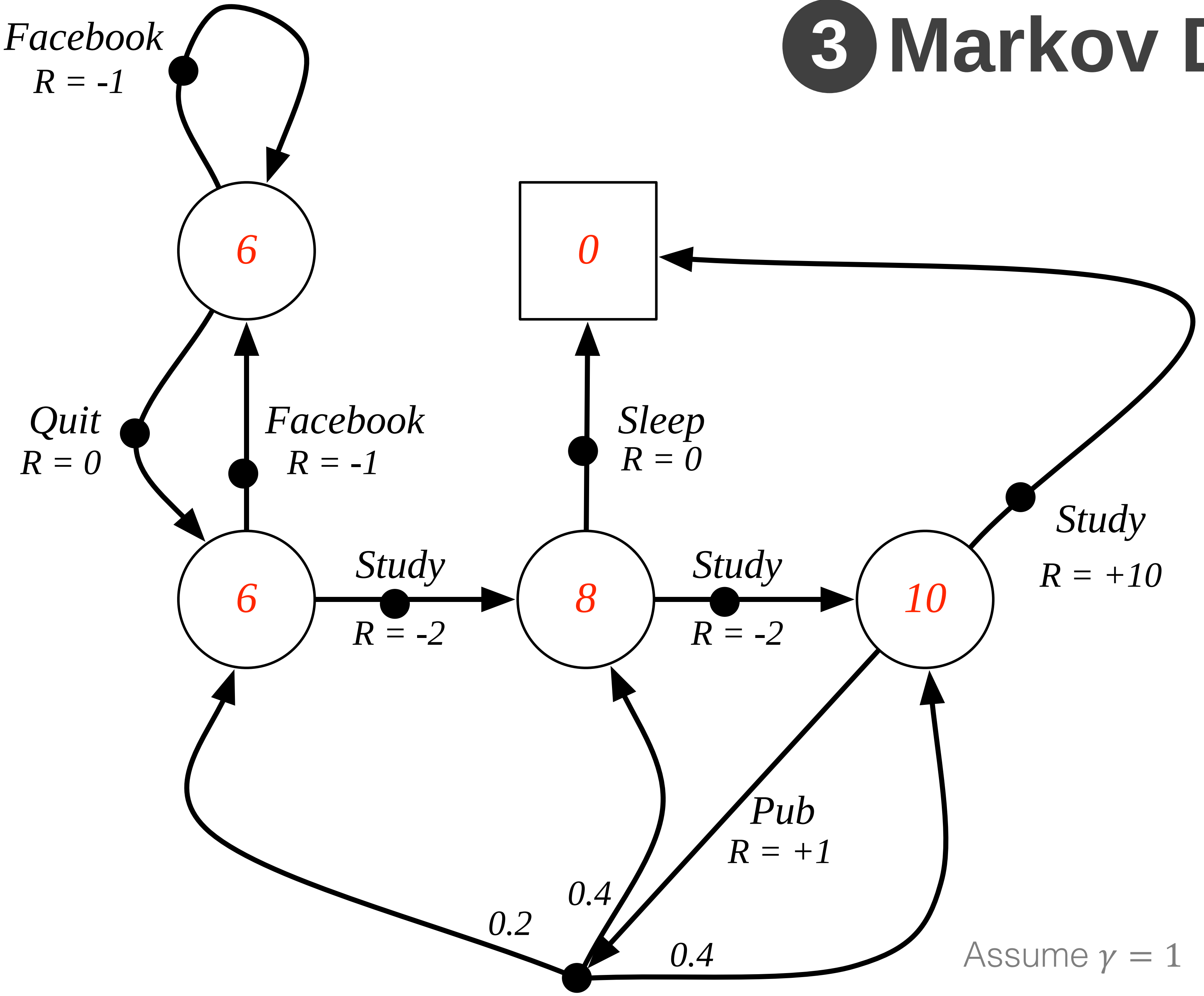
$$v_\pi(s) = E_\pi[R_{t+1} + \gamma v_\pi(S_{t+1})|s]$$

$$-1.3 \approx 0.5\times(-1 - 2.3) + 0.5\times(-2 + 2.7)$$

Assume $\gamma = 1$

Example from David Silver, UCL, 2015

❸ Markov Decision Process



Optimal **state-value** function, $v_*(s)$

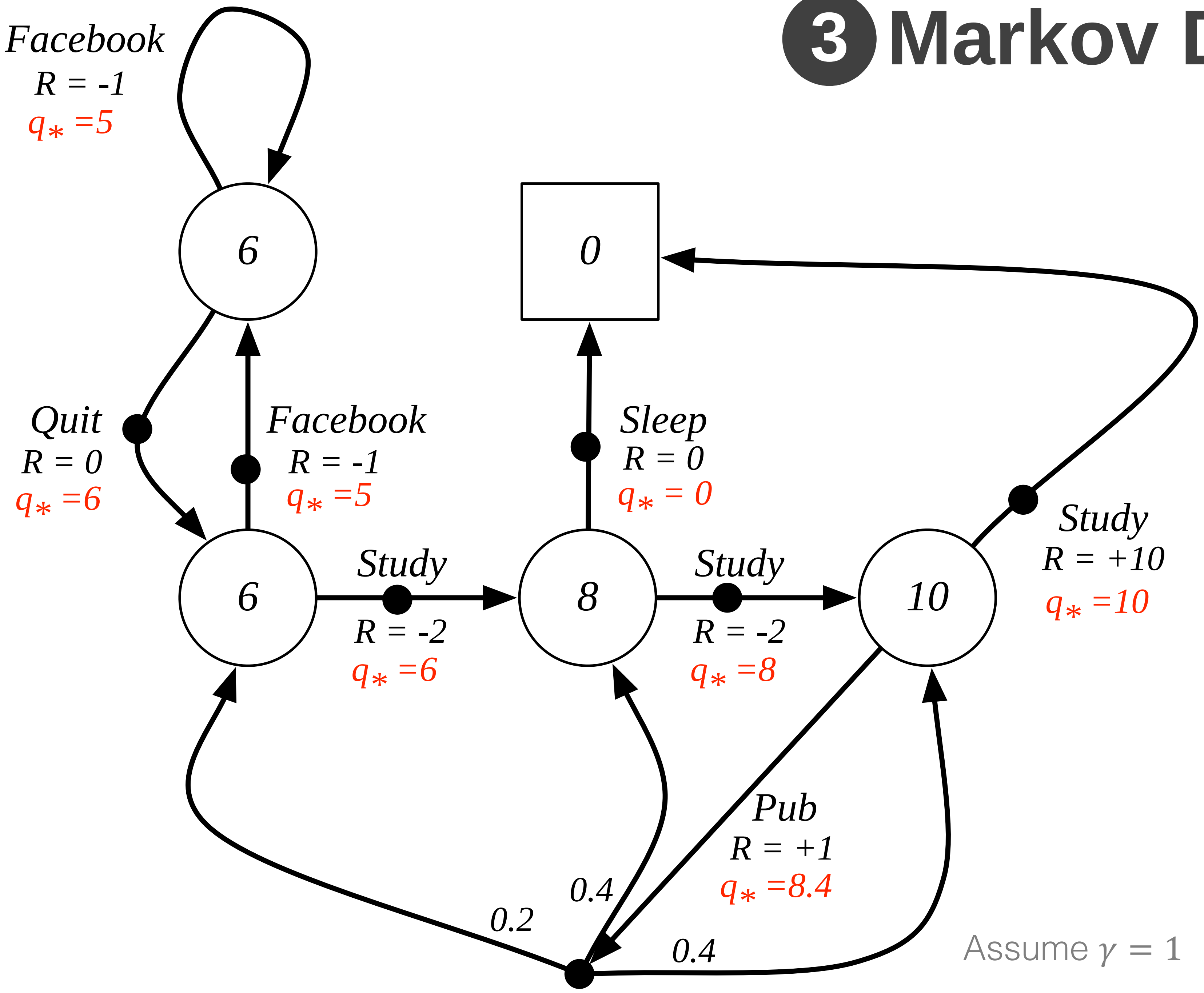
Maximum value function over all policies

$$v_*(s) = \max_\pi v_\pi(s)$$

Assume $\gamma = 1$

Example from David Silver, UCL, 2015

3 Markov Decision Process



Assume $\gamma = 1$

Optimal **state-value** function, $v_*(s)$

Maximum value function over all policies

$$v_*(s) = \max_\pi v_\pi(s)$$

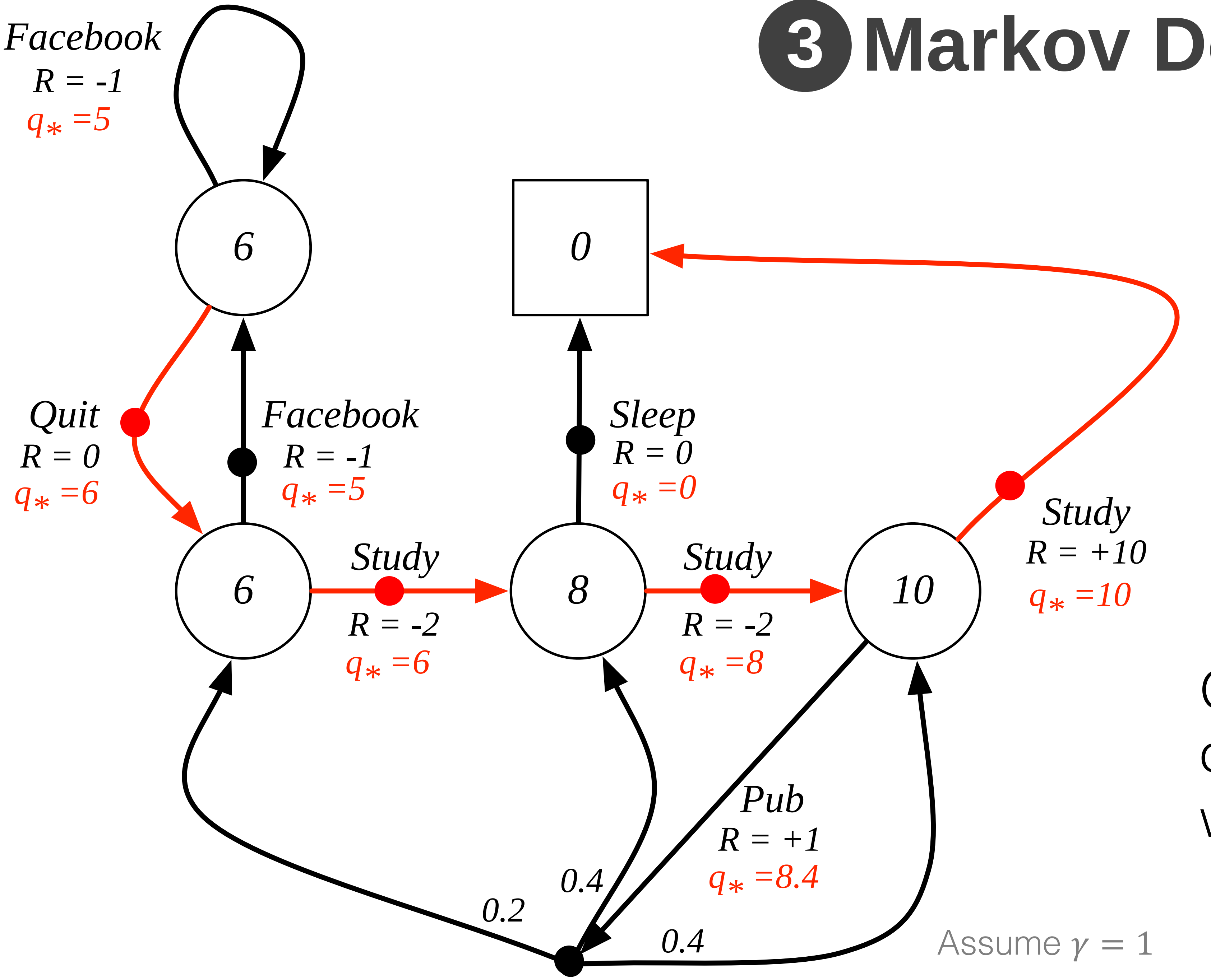
Optimal **action-value** function, $q_*(s,a)$

Maximum value function over all policies

$$q_*(s,a) = \max_\pi q_\pi(s,a)$$

Example from David Silver, UCL, 2015

3 Markov Decision Process



Optimal **policy**, $\pi_*(s)$

Which action to take at each moment

$$\pi_*(s) = \arg\max_{a} q_*(s, a)$$

Once we have the optimal value functions, we've "solved" the MDP!

Assume $\gamma = 1$

Example from David Silver, UCL, 2015

Building blocks for the full RL problem

1	Markov Chain	{state space S, transition probabilities P}
2	Markov Reward Process (MRP)	{S, P, + rewards R, discount rate γ} adds rewards (and values)
3	Markov Decision Process (MDP)	{S, P, R, γ, + actions A} adds decisions (i.e. the ability to control)

- **RL methods do NOT ASSUME knowledge of P or R (while dynamic programming does)**
- **RL learns/approximates that knowledge**

Adapted from David Silver, 2015

Markov Decision Process

If we know the components of the MDP, we can use those to calculate the value functions and determine the optimal policy

Components:
State space S
Transition probabilities, P
Rewards, R
Discount rate, γ
Actions, A

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)

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

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

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

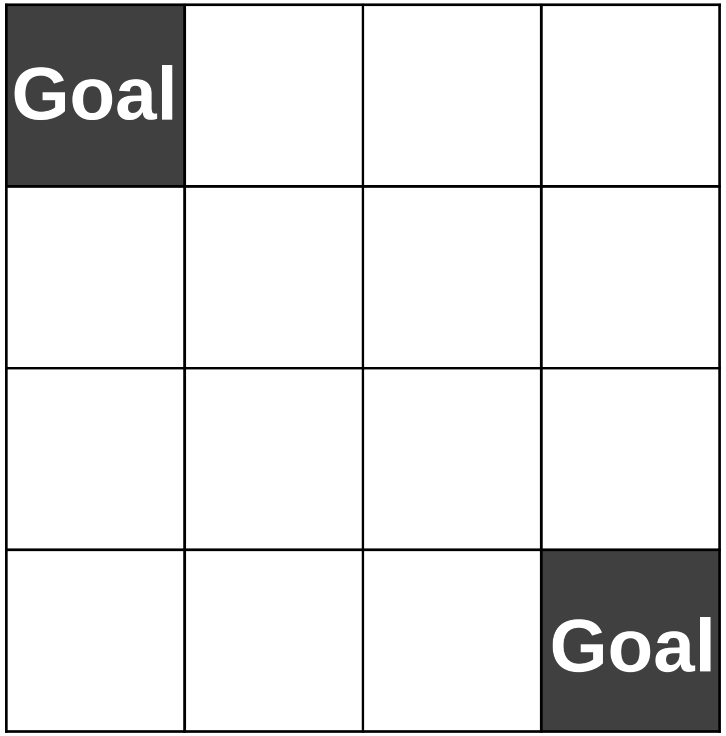
Roadmap to optimal policies

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

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

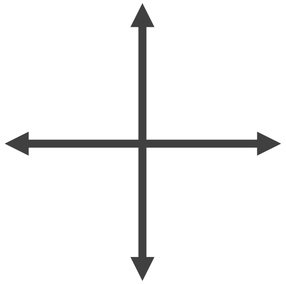
What if we don't have a fully known MDP? **Monte Carlo Methods**

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

Dynamic Programming

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

$v_0(s)$

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:

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

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)

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

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

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

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

$v_0(s)$

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**	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**	

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

Value function:

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

$v_0(s)$

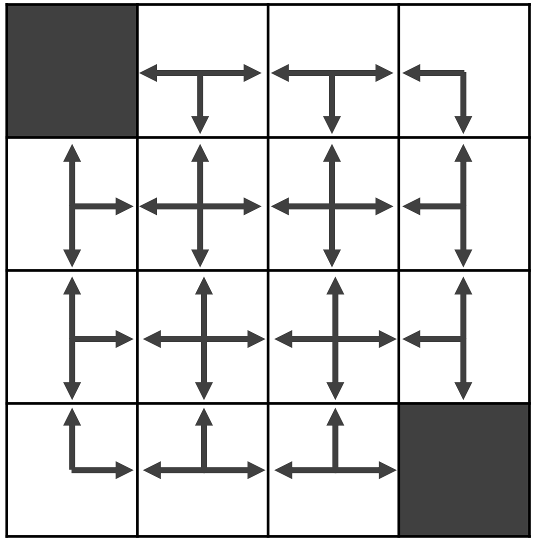
0	0	0	0
0	0	0	0
0	0	0	0
0	0	0	0

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

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

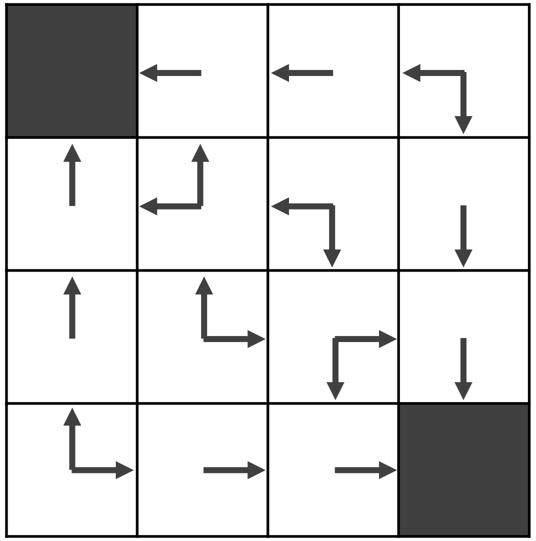
Initial policy: $\pi(s)$

$\pi(a|s) =$ 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**
2. Find a **better** policy? **Policy improvement**
3. Find the **best** policy? **Policy iteration**
4. Find the best policy **faster**? **Value iteration**

Dynamic Programming

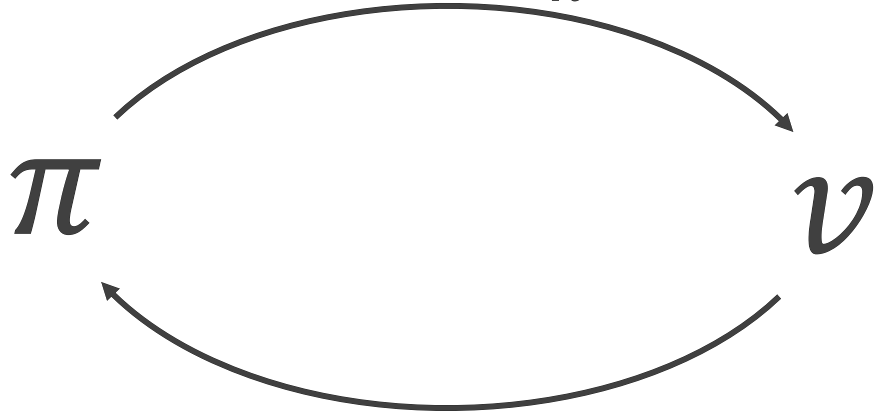
What if we don't have a fully known MDP? **Monte Carlo Methods**

3. Policy Iteration

Find the **best** policy

Policy **Evaluation**

$v \rightarrow v_\pi$



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

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**

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

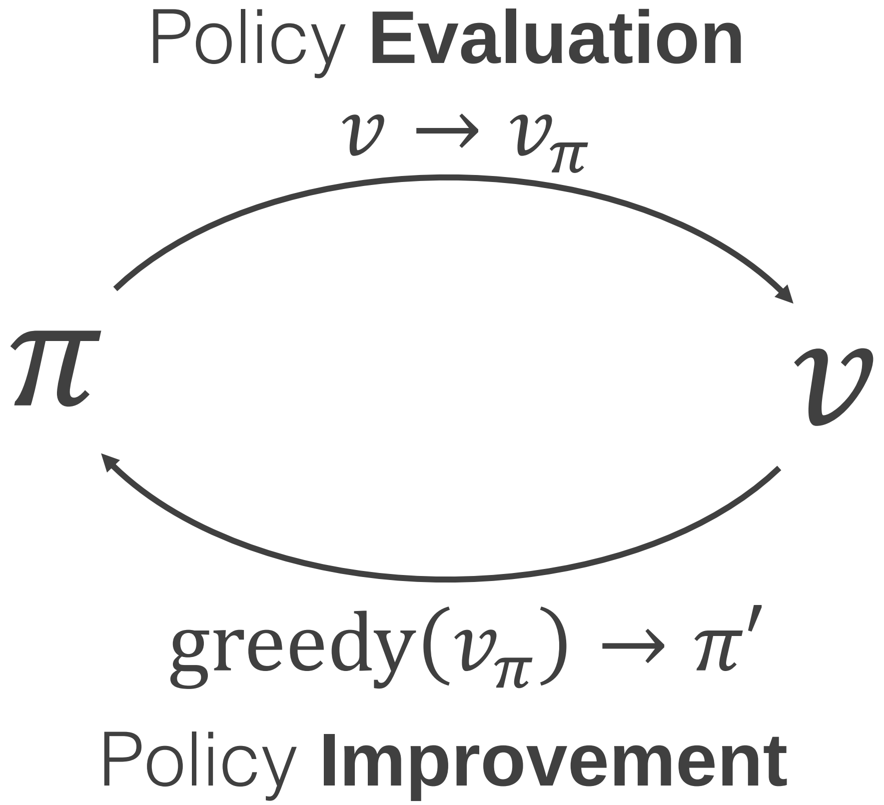
Adapted from David Silver, 2015 and Sutton and Barto, 1998

3. Policy Iteration

Find the **best** policy

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

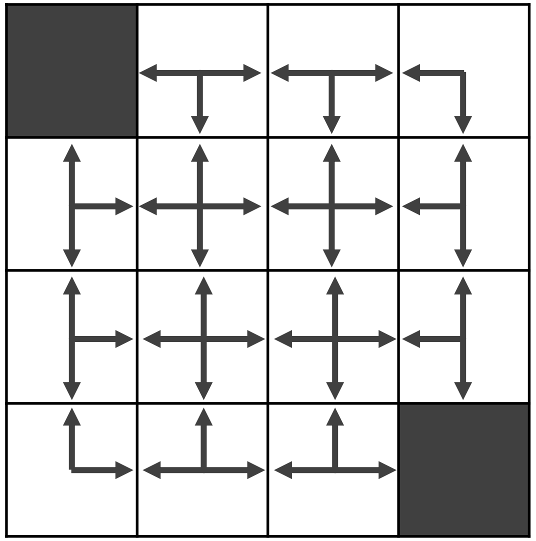
Output: **best** policy $\pi^*(a|s)$



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 $\pi_0(s)$

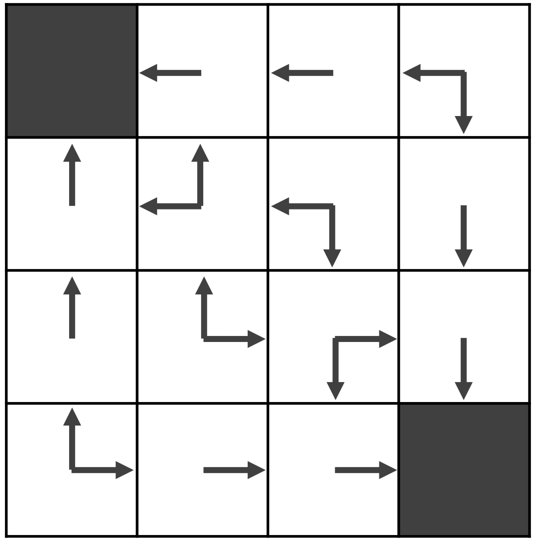


Policy Evaluation

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

Policy Improvement

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



Policy Evaluation

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

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

$v_0(s)$

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

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

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

$v_0(s)$

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

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

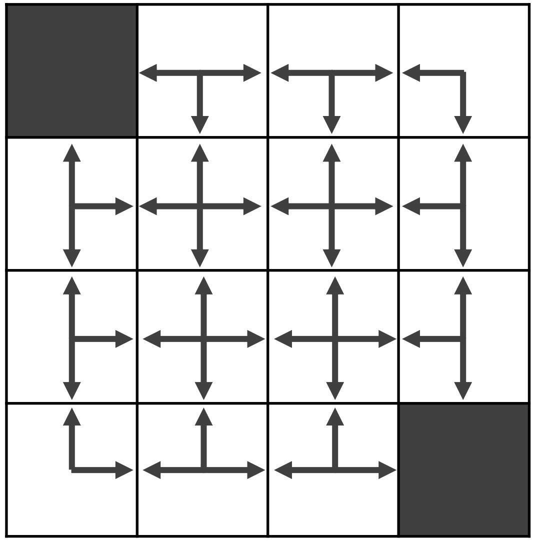
0	-1	-2	-3
-1	-2	-3	-2
-2	-3	-2	-1
-3	-2	-1	0

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**	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**	

$\pi_0(s)$



$v_0(s)$

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

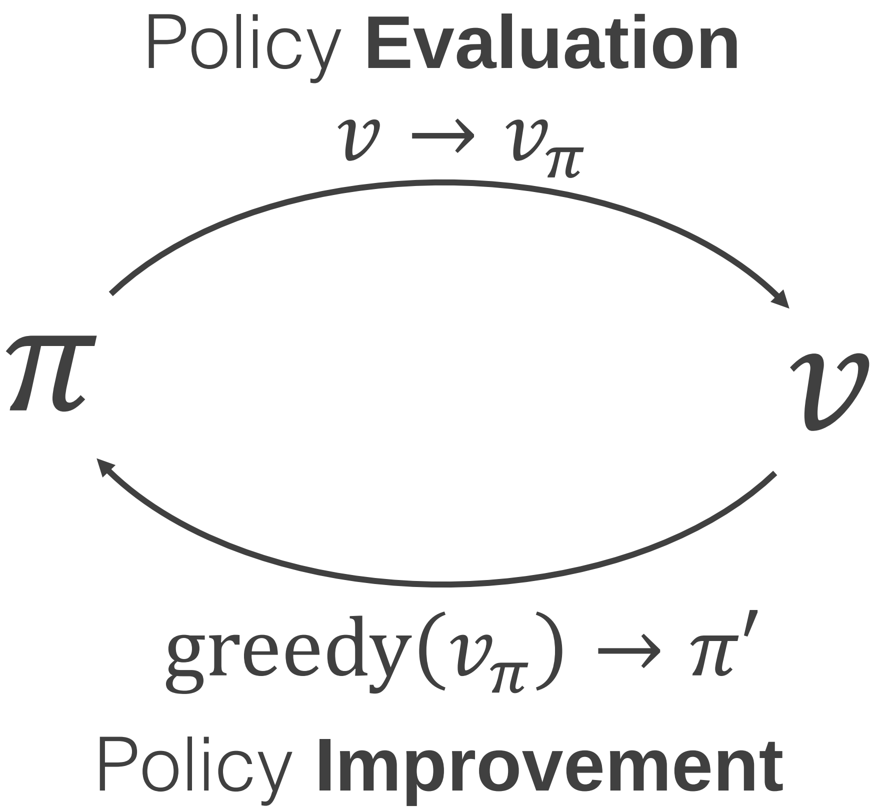
Find the best policy **faster**

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

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

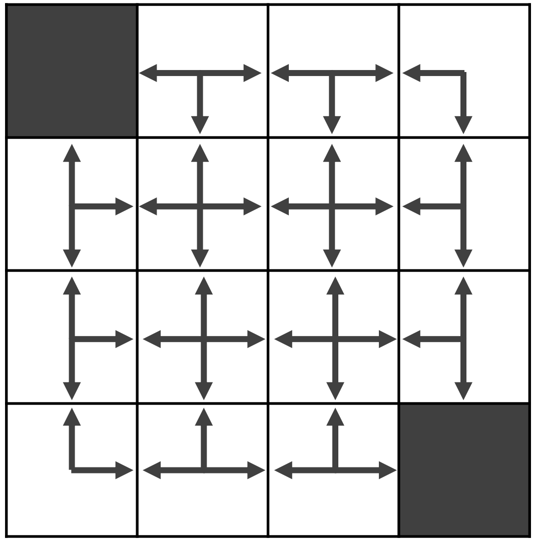
0	-14	-20	-22
-14	-18	-20	-20
-20	-20	-18	-14
-22	-20	-14	

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

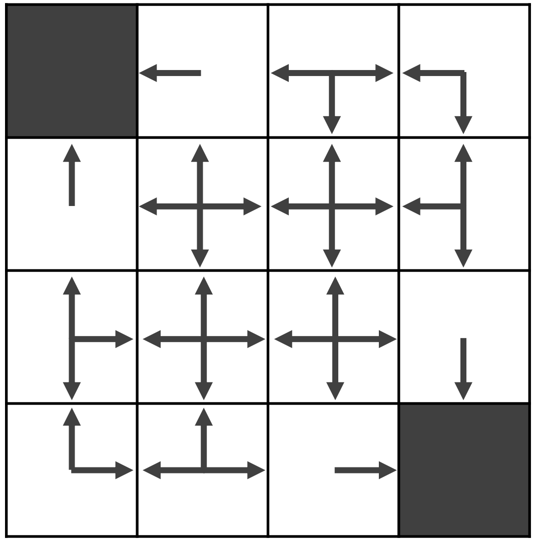
2 $\pi_0(s)$



3 $v_1(s)$

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

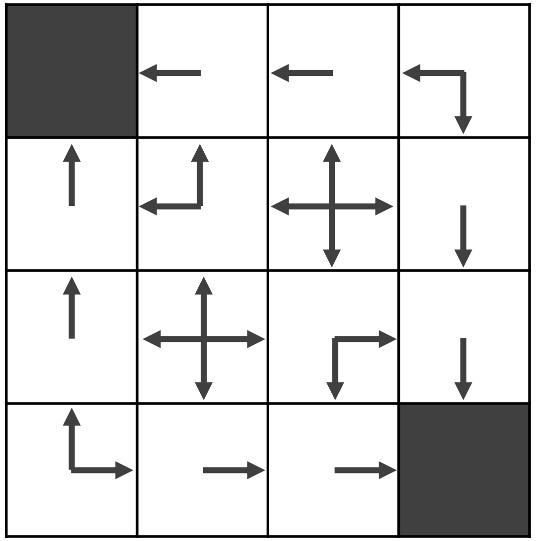
4 $\pi_1(s)$



5 $v_2(s)$

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

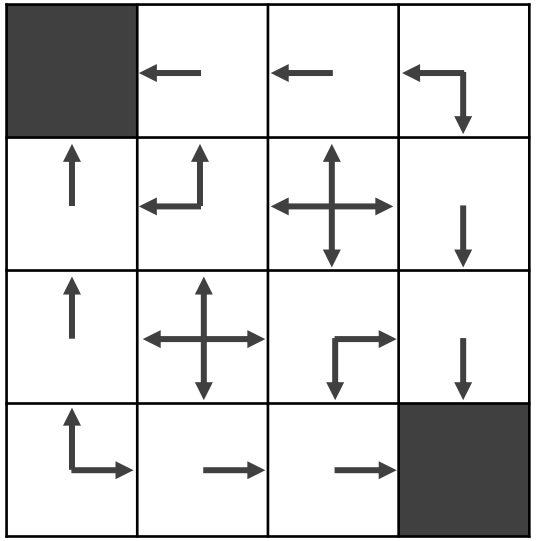
6 $\pi_2(s)$



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

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

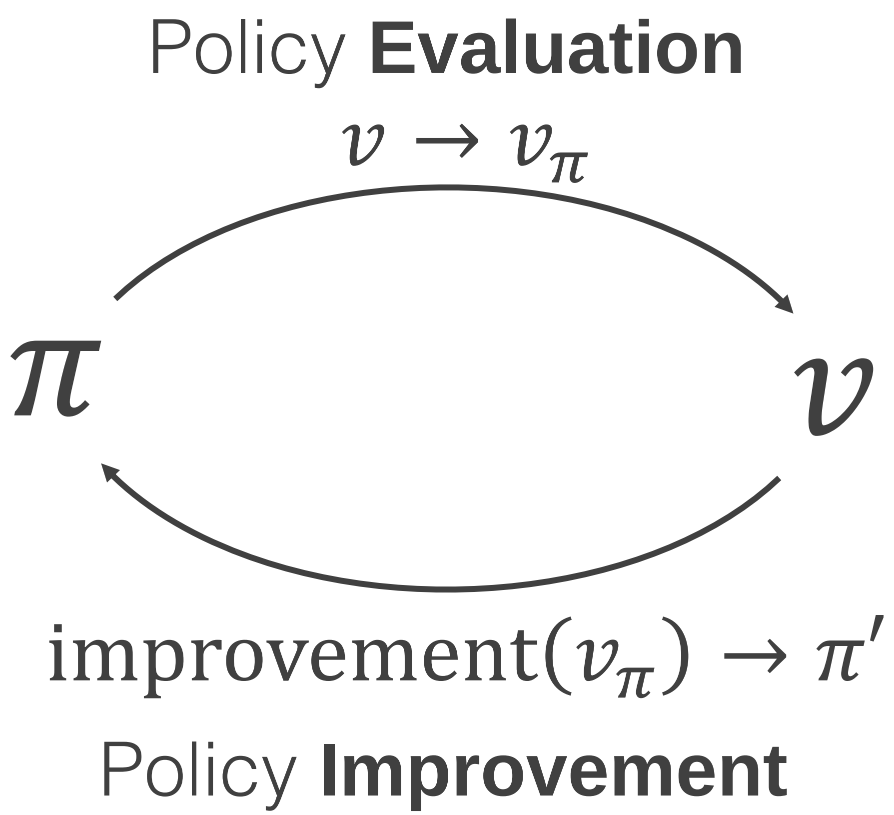
8 $\pi_3(s) = \pi^*(s)$



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**!

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?