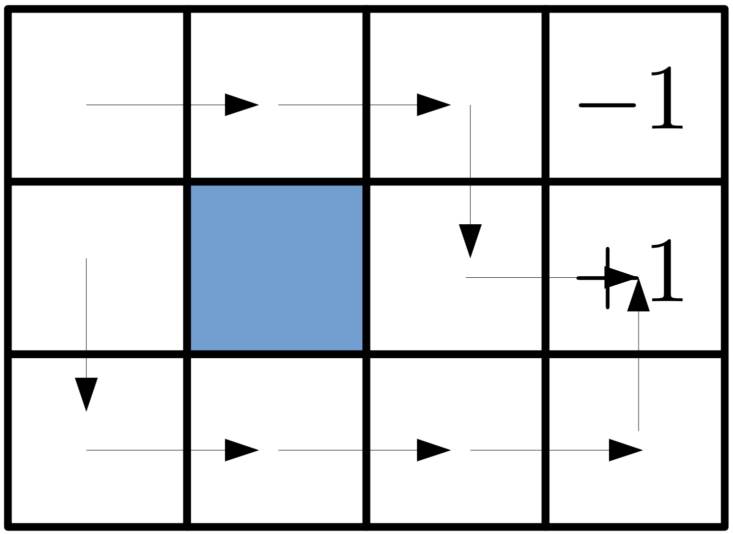
Dynamic Programming

Robert Platt
Northeastern University

Recall: Policy



A *policy* is a rule for selecting actions: $\pi : S \rightarrow A$

$$\pi(s) = a$$

If agent is in *this* state, then take *this* action

A policy can be stochastic: $\pi(a|s) = P(a_t = a|s_t = s)$

<u>The goal of this lecture is to develop new ways of calculating an optimal policy</u>

Calculating an optimal policy

The goal of this lecture is to develop new ways of calculating an optimal policy

– first, develop methods of calculating value function for an arbitrary policy (policy evaluation)

– then, develop methods of calculating an optimal value function (and policy) by iteratively calculating value function and then improving policy

Recall: Value Function

Value of state s when acting according to policy π :

$$V^{\pi}(s) = \mathbb{E}_{\pi}[G_t | s_t = s]$$

Value of a state == expected return from that state
if agent follows policy π

Recall: Value Function

Value of state s when acting according to policy π :

$$V^{\pi}(s) = \mathbb{E}_{\pi}[G_t | s_t = s]$$

Value of a state == expected return from that state
if agent follows policy π

But, how do we compute the value function?
(for a particular policy)

Recall: Value Function

Value of state s when acting according to policy π :

$$V^\pi(s) = \mathbb{E}_\pi[G_t | s_t = s]$$

Value of a state == expected return from that state if agent follows policy π

But, how do we compute the value function?
(for a particular policy)

Possible methods:

1. monte carlo methods

Recall: Value Function

Value of state s when acting according to policy π :

$$V^\pi(s) = \mathbb{E}_\pi[G_t | s_t = s]$$

Value of a state == expected return from that state
if agent follows policy π

But, how do we compute the value function?
(for a particular policy)

<u>Possible methods:</u>

1. monte carlo methods
2. dynamic programming

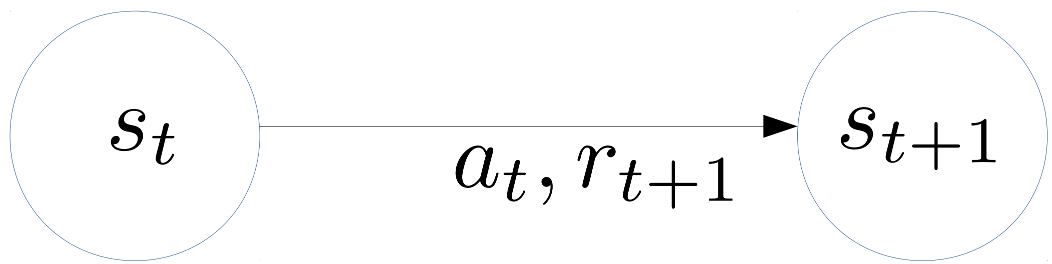
New method that will be introduced today

Policy Evaluation

How do we calculate the value function for a given policy?

Policy Evaluation

How do we calculate the value function for a given policy?



$$V^\pi(s_t = s) = ?$$

Policy Evaluation

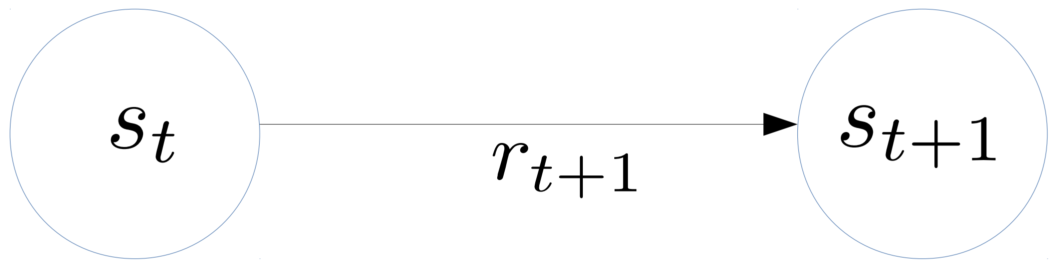
How do we calculate the value function for a given policy?

$$V^\pi(s_t = s) = ?$$

For simplicity, let’s ignore action for now

Policy Evaluation

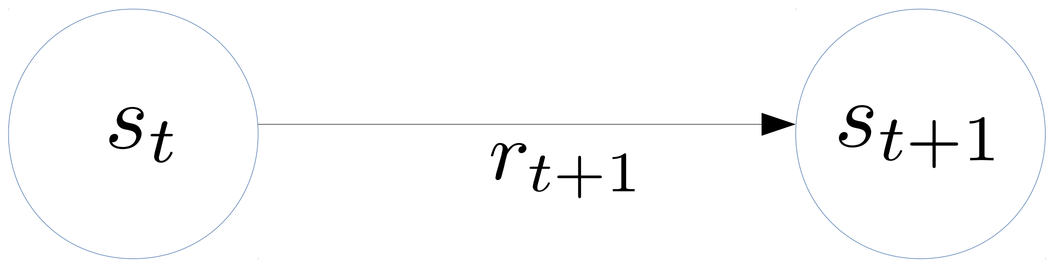
How do we calculate the value function for a given policy?



$$V^{\pi}(s_t = s) = ?$$

Policy Evaluation

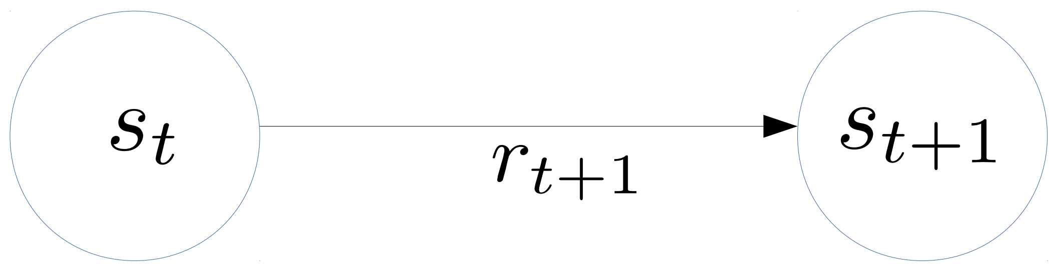
How do we calculate the value function for a given policy?



$$V^\pi(s_t = s) = r_{t+1} + \text{expected value of being at } s_{t+1}$$

Policy Evaluation

How do we calculate the value function for a given policy?

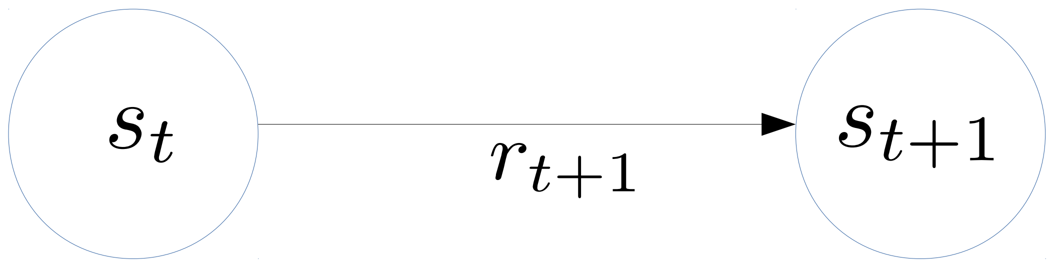


$$V^{\pi}(s_t = s) = r_{t+1} + \underbrace{\text{expected value of being at } s_{t+1}}$$

Another expression for this?

Policy Evaluation

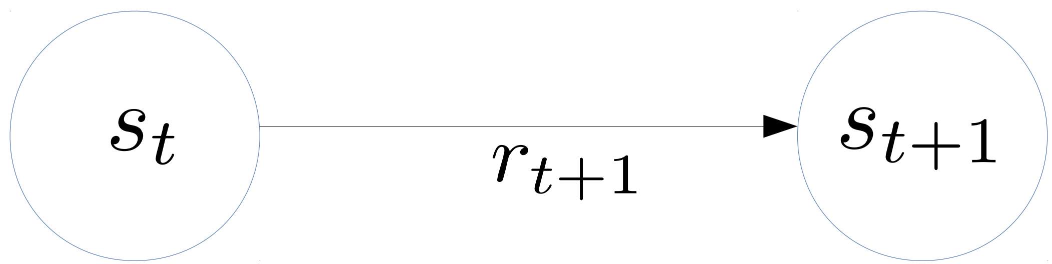
How do we calculate the value function for a given policy?



$$V^\pi(s_t = s) = r_{t+1} + \underbrace{\text{expected value of being at } s_{t+1}}_{\gamma V^\pi(s_{t+1} = s')}$$

Policy Evaluation

How do we calculate the value function for a given policy?

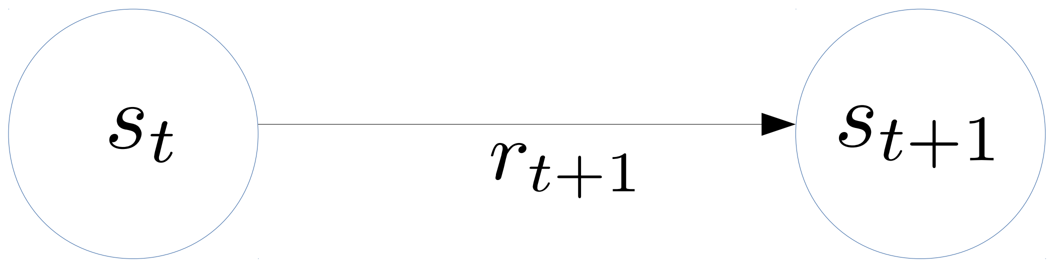


$$V^\pi(s_t = s) = r_{t+1} + \underbrace{\text{expected value of being at } s_{t+1}}_{\gamma V^\pi(s_{t+1} = s')}$$

Why is there a gamma here?

Policy Evaluation

How do we calculate the value function for a given policy?

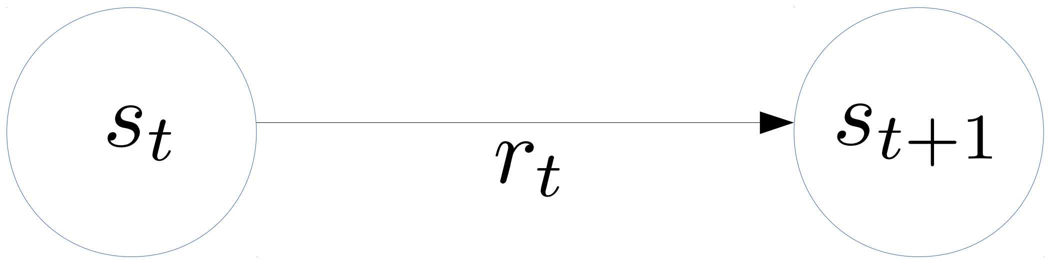


$$V^\pi(s_t = s) = r_{t+1} + \underbrace{\text{expected value of being at } s_{t+1}}_{\gamma V^\pi(s_{t+1} = s')}$$

$$V^\pi(s_t = s) = \mathbb{E}_\pi[r_{t+1} + \gamma V^\pi(s_{t+1} = s') | s_t = s]$$

Policy Evaluation

How do we calculate the value function for a given policy?



$$V^\pi(s_t = s) = r_{t+1} + \text{expected value of being at } s_{t+1}$$

$$= \mathbb{E}_\pi[r_{t+1} + \gamma V^\pi(s_{t+1} = s')|s_t = s]$$

Think-pair-share

Please write this expectation in terms of:

$p(s', r|s, a)$ == prob of s',r given s,a

$\pi(s)$ == action to select from state *s*

y?

$V^\pi(s_t = s)$ ... $_{+1}$ + expected value of being at s_{t+1}

$$= \mathbb{E}_\pi[r_{t+1} + \gamma V^\pi(s_{t+1} = s')|s_t = s]$$

$$=$$

Policy Evaluation

For stochastic action selection:

$p(s', r|s, a)$ == prob of s',r given s,a

$\pi(a|s)$ == prob of selecting action a from state s

y?

$$V^{\pi}(s_t = s) \quad {}_{+1} + \text{expected value of being at } s_{t+1}$$

$$= \mathbb{E}_{\pi}[r_{t+1} + \gamma V^{\pi}(s_{t+1} = s')|s_t = s]$$

$$= \sum_{a} \pi(a|s) \sum_{s',r} p(s', r|s, a)[r + \gamma V^{\pi}(s_{t+1} = s')]$$

Policy Evaluation

For stochastic action selection:

$p(s', r|s, a)$ == prob of s’,r given s,a

$\pi(a|s)$ == prob of selecting action a from state s

y?

$V^{\pi}(s_t = s)$ ... $_{+1}$ + expected value of being at s_{t+1}

$$= \mathbb{E}_{\pi}[r_{t+1} + \gamma V^{\pi}(s_{t+1} = s')|s_t = s]$$

$$= \sum_{a} \pi(a|s) \sum_{s',r} p(s', r|s, a)[r + \gamma V^{\pi}(s_{t+1} = s')]$$

Or, more simply: $V^{\pi}(s) = \sum_{a} \pi(a|s) \sum_{s',r} p(s', r|s, a)[r + \gamma V^{\pi}(s')]$

(SB, eqn 4.4)

Policy Evaluation

For stochastic action selection:

$p(s', r|s, a)$ == prob of s',r given s,a

$\pi(a|s)$ == prob of selecting action a from state s

y?

$$V^\pi(s_t = s) \quad {}_{+1} + \text{expected value of being at } s_{t+1}$$

$$= \mathbb{E}_\pi[r_{t+1} + \gamma V^\pi(s_{t+1} = s')|s_t = s]$$

$$= \sum_a \pi(a|s) \sum_{s',r} p(s', r|s, a)[r + \gamma V^\pi(s_{t+1} = s')]$$

Called the *Bellman Equation*

Or, more simply: $$V^\pi(s) = \sum_a \pi(a|s) \sum_{s',r} p(s', r|s, a)[r + \gamma V^\pi(s')]$$

(SB, eqn 4.4)

Policy Evaluation Algorithm

Iterative policy evaluation, SB pp 61

Input π, the policy to be evaluated
Initialize an array $V(s) = 0$, for all $s \in \mathcal{S}^+$
Repeat
    $\Delta \leftarrow 0$
    For each $s \in \mathcal{S}$:
        $v \leftarrow V(s)$
        $V(s) \leftarrow \sum_a \pi(a|s) \sum_{s',r} p(s', r \,|\, s, a)\big[r + \gamma V(s')\big]$
        $\Delta \leftarrow \max(\Delta, |v - V(s)|)$
until $\Delta < \theta$ (a small positive number)
Output $V \approx v_\pi$

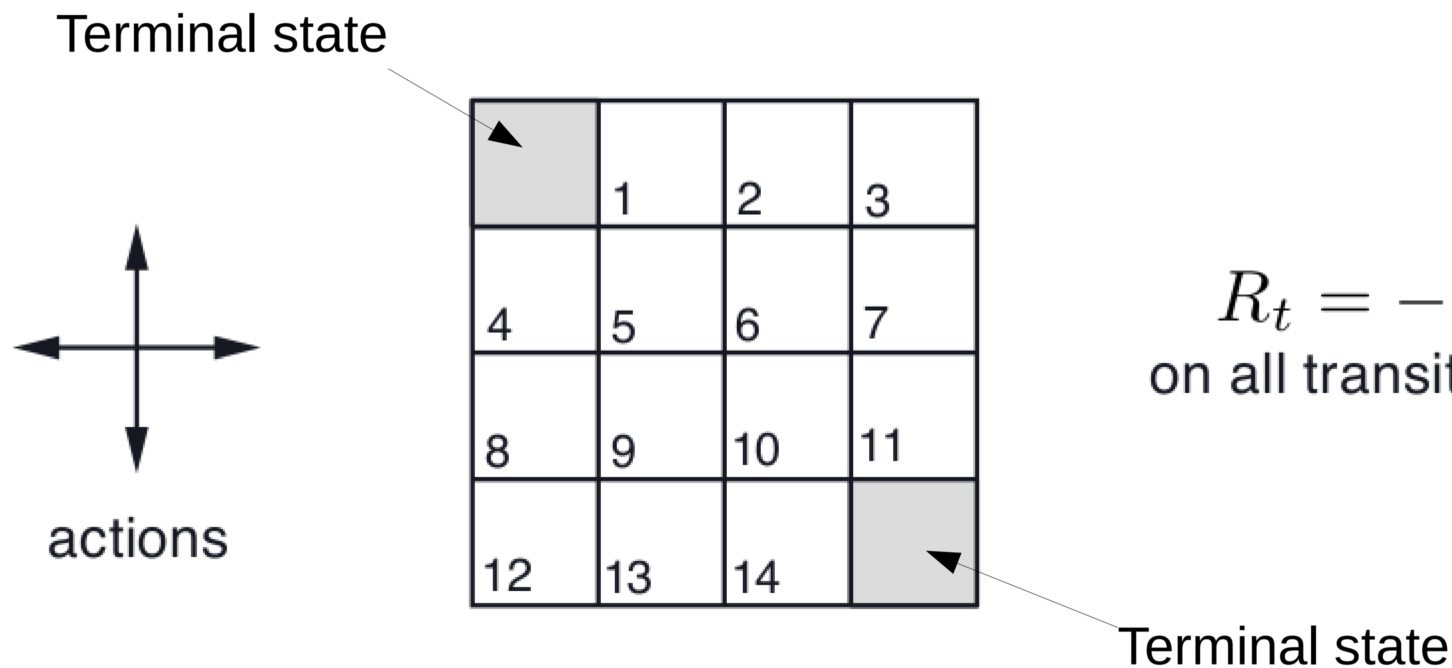
Policy Evaluation Algorithm

Iterative policy evaluation, SB pp 61

Input π, the policy to be evaluated
Initialize an array $V(s) = 0$, for all $s \in \mathcal{S}^+$
Repeat
  $\Delta \leftarrow 0$
  For each $s \in \mathcal{S}$:
    $v \leftarrow V(s)$
    $V(s) \leftarrow \sum_a \pi(a|s) \sum_{s',r} p(s', r \,|\, s, a)\big[r + \gamma V(s')\big]$
    $\Delta \leftarrow \max(\Delta, |v - V(s)|)$
until $\Delta < \theta$ (a small positive number)
Output $V \approx v_\pi$

Bellman Equation

Policy Evaluation Algorithm: SB example 4.1

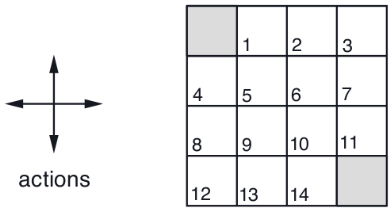


$R_t = -1$
on all transitions

$$\mathcal{S} = \{1, \ldots, 14\}$$
$$\mathcal{A} = \{left, right, up, down\}$$

State transitions: deterministic
Undiscounted

Policy Evaluation Algorithm: SB example 4.1



$R_t = -1$
on all transitions

Initialize value function at zero

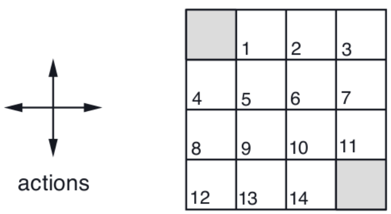
0.0	0.0	0.0	0.0
0.0	0.0	0.0	0.0
0.0	0.0	0.0	0.0
0.0	0.0	0.0	0.0

Evaluate *V* for a policy that selects actions uniformly randomly

Policy Evaluation Algorithm: SB example 4.1

0.0	0.0	0.0	0.0
0.0	0.0	0.0	0.0
0.0	0.0	0.0	0.0
0.0	0.0	0.0	0.0

<u>What does this value become on first iteration?</u>



$R_t = -1$
on all transitions

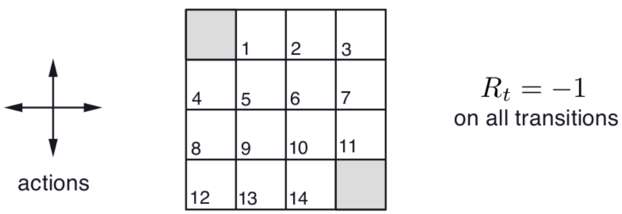
Input π, the policy to be evaluated
Initialize an array $V(s) = 0$, for all $s \in \mathcal{S}^+$
Repeat
  $\Delta \leftarrow 0$
  For each $s \in \mathcal{S}$:
    $v \leftarrow V(s)$
    $V(s) \leftarrow \sum_a \pi(a|s) \sum_{s',r} p(s', r|s, a)\big[r + \gamma V(s')\big]$
    $\Delta \leftarrow \max(\Delta, |v - V(s)|)$
until $\Delta < \theta$ (a small positive number)
Output $V \approx v_\pi$

Policy Evaluation Algorithm: SB example 4.1

0.0	0.0	0.0	0.0
0.0	0.0	0.0	0.0
0.0	0.0	0.0	0.0
0.0	0.0	0.0	0.0

<u>What does this value become on first iteration?</u>

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



Input π, the policy to be evaluated
Initialize an array $V(s) = 0$, for all $s \in \mathcal{S}^+$
Repeat
  $\Delta \leftarrow 0$
  For each $s \in \mathcal{S}$:
    $v \leftarrow V(s)$
    $V(s) \leftarrow \sum_a \pi(a|s) \sum_{s',r} p(s',r|s,a)\big[r + \gamma V(s')\big]$
    $\Delta \leftarrow \max(\Delta, |v - V(s)|)$
until $\Delta < \theta$ (a small positive number)
Output $V \approx v_\pi$

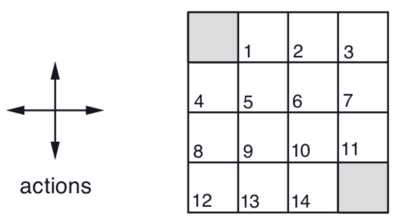
Policy Evaluation Algorithm: SB example 4.1

0.0	0.0	0.0	0.0
0.0	0.0	0.0	0.0
0.0	0.0	0.0	0.0
0.0	0.0	0.0	0.0

<u>What does this value become on first iteration?</u>

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

$$= \sum_a \pi(a|s)[r + \gamma V^\pi(s')] \quad \text{(b/c deterministic)}$$

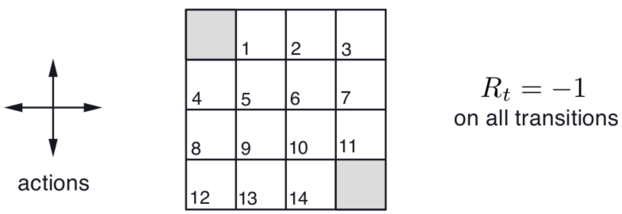


$R_t = -1$
on all transitions

Input π, the policy to be evaluated
Initialize an array $V(s) = 0$, for all $s \in \mathcal{S}^+$
Repeat
  $\Delta \leftarrow 0$
  For each $s \in \mathcal{S}$:
    $v \leftarrow V(s)$
    $V(s) \leftarrow \sum_a \pi(a|s) \sum_{s',r} p(s',r|s,a)\big[r + \gamma V(s')\big]$
    $\Delta \leftarrow \max(\Delta, |v - V(s)|)$
until $\Delta < \theta$ (a small positive number)
Output $V \approx v_\pi$

Policy Evaluation Algorithm: SB example 4.1

0.0	0.0	0.0	0.0
0.0	0.0	0.0	0.0
0.0	0.0	0.0	0.0
0.0	0.0	0.0	0.0



What does this value become on first iteration?

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

$$= \sum_a \pi(a|s)[r + \gamma V^\pi(s')] \quad \text{(b/c deterministic)}$$

$$= \sum_a 0.25[-1 + \gamma V^\pi(s')]$$

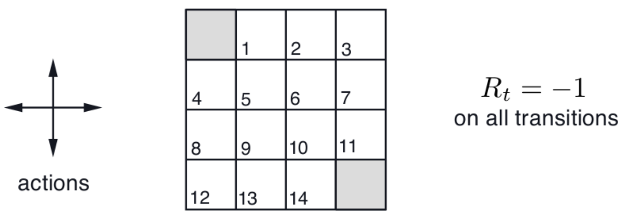
$$= \sum_a 0.25[-1 + \gamma 0]$$

$$= -1$$

Input π, the policy to be evaluated
Initialize an array $V(s) = 0$, for all $s \in \mathcal{S}^+$
Repeat
  $\Delta \leftarrow 0$
  For each $s \in \mathcal{S}$:
    $v \leftarrow V(s)$
    $V(s) \leftarrow \sum_a \pi(a|s) \sum_{s',r} p(s',r|s,a)\big[r + \gamma V(s')\big]$
    $\Delta \leftarrow \max(\Delta, |v - V(s)|)$
until $\Delta < \theta$ (a small positive number)
Output $V \approx v_\pi$

Policy Evaluation Algorithm: SB example 4.1

0.0	-1.0	-1.0	-1.0
-1.0	-1.0	-1.0	-1.0
-1.0	-1.0	-1.0	-1.0
-1.0	-1.0	-1.0	0.0



What does this value become on first iteration?

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

$$= \sum_a \pi(a|s)[r + \gamma V^\pi(s')] \quad \text{(b/c deterministic)}$$

$$= \sum_a 0.25[-1 + \gamma V^\pi(s')]$$

$$= \sum_a 0.25[-1 + \gamma 0]$$

$$= -1$$

Input π, the policy to be evaluated
Initialize an array $V(s) = 0$, for all $s \in \mathcal{S}^+$
Repeat
  $\Delta \leftarrow 0$
  For each $s \in \mathcal{S}$:
    $v \leftarrow V(s)$
    $V(s) \leftarrow \sum_a \pi(a|s) \sum_{s',r} p(s',r|s,a)\big[r + \gamma V(s')\big]$
    $\Delta \leftarrow \max(\Delta, |v - V(s)|)$
until $\Delta < \theta$ (a small positive number)
Output $V \approx v_\pi$

Policy Evaluation Algorithm: SB example 4.1

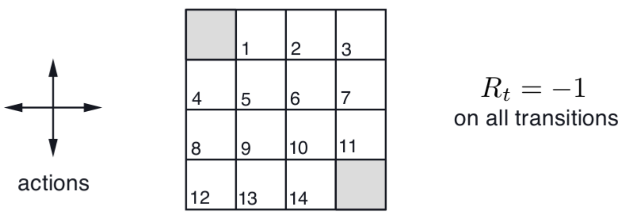
0.0	-1.0	-1.0	-1.0
-1.0	-1.0	-1.0	-1.0
-1.0	-1.0	-1.0	-1.0
-1.0	-1.0	-1.0	0.0

<u>What does this value become on second iteration?</u>

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

$$= \sum_a \pi(a|s)[r + \gamma V^\pi(s')] \quad \text{(b/c deterministic)}$$

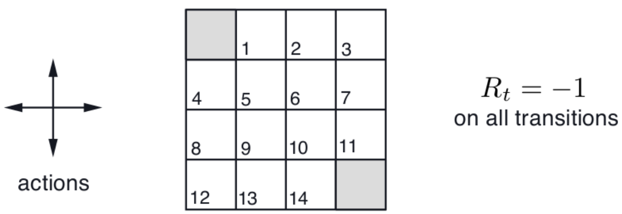
$$= \sum_a 0.25[-1 + \gamma V^\pi(s')]$$



Input π, the policy to be evaluated
Initialize an array $V(s) = 0$, for all $s \in \mathcal{S}^+$
Repeat
  $\Delta \leftarrow 0$
  For each $s \in \mathcal{S}$:
    $v \leftarrow V(s)$
    $V(s) \leftarrow \sum_a \pi(a|s) \sum_{s',r} p(s',r|s,a)\left[r + \gamma V(s')\right]$
    $\Delta \leftarrow \max(\Delta, |v - V(s)|)$
until $\Delta < \theta$ (a small positive number)
Output $V \approx v_\pi$

Policy Evaluation Algorithm: SB example 4.1

0.0	-1.0	-1.0	-1.0
-1.0	-1.0	-1.0	-1.0
-1.0	-1.0	-1.0	-1.0
-1.0	-1.0	-1.0	0.0



What does this value become on second iteration?

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

$$= \sum_a \pi(a|s)[r + \gamma V^\pi(s')] \quad \text{(b/c deterministic)}$$

$$= \sum_a 0.25[-1 + \gamma V^\pi(s')]$$

$$= -\frac{1}{4} - \frac{2}{4} - \frac{2}{4} - \frac{2}{4}$$

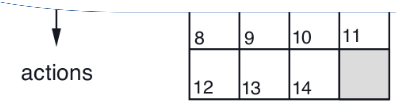
$$= -1.75$$

Input π, the policy to be evaluated
Initialize an array $V(s) = 0$, for all $s \in \mathcal{S}^+$
Repeat
  $\Delta \leftarrow 0$
  For each $s \in \mathcal{S}$:
    $v \leftarrow V(s)$
    $V(s) \leftarrow \sum_a \pi(a|s) \sum_{s',r} p(s', r|s, a)\left[r + \gamma V(s')\right]$
    $\Delta \leftarrow \max(\Delta, |v - V(s)|)$
until $\Delta < \theta$ (a small positive number)
Output $V \approx v_\pi$

Policy Evaluation Algorithm: SB example 4.1

0.0	-1.7	-2.0	-2.0
-1.7	-2.0	-2.0	-2.0
-2.0	-2.0	-2.0	-1.7
-2.0	[illegible]	[illegible]	0.0

Why is this value NOT -1.75?



What does this value become on second iteration?

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

$$= \sum_a \pi(a|s)[r + \gamma V^\pi(s')]$$ (b/c deterministic)

$$= \sum_a 0.25[-1 + \gamma V^\pi(s')]$$

$$= -\frac{1}{4} - \frac{2}{4} - \frac{2}{4} - \frac{2}{4}$$

$$= -1.75$$

Input π, the policy to be evaluated
Initialize an array $V(s) = 0$, for all $s \in \mathcal{S}^+$
Repeat
  $\Delta \leftarrow 0$
  For each $s \in \mathcal{S}$:
    $v \leftarrow V(s)$
    $V(s) \leftarrow \sum_a \pi(a|s) \sum_{s',r} p(s',r|s,a)\big[r + \gamma V(s')\big]$
    $\Delta \leftarrow \max(\Delta, |v - V(s)|)$
until $\Delta < \theta$ (a small positive number)
Output $V \approx v_\pi$

Policy Evaluation Algorithm: SB example 4.1

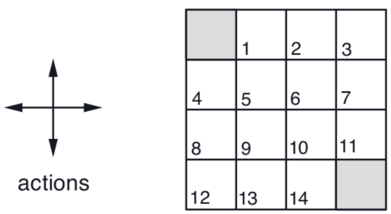
0.0	-1.7	-2.0	-2.0
-1.7	-2.0	-2.0	-2.0
-2.0	-2.0	-2.0	-1.7
-2.0	-2.0	-1.7	0.0

<u>What does this value become on third iteration?</u>

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

$$= \sum_a \pi(a|s)[r + \gamma V^\pi(s')]$$ (b/c deterministic)

$$=$$

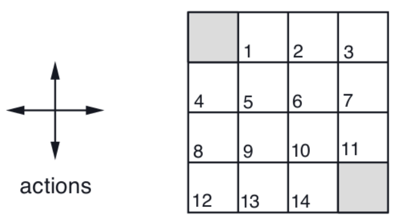


$R_t = -1$
on all transitions

Input π, the policy to be evaluated
Initialize an array $V(s) = 0$, for all $s \in \mathcal{S}^+$
Repeat
$\Delta \leftarrow 0$
For each $s \in \mathcal{S}$:
$v \leftarrow V(s)$
$V(s) \leftarrow \sum_a \pi(a|s) \sum_{s',r} p(s',r|s,a)\big[r + \gamma V(s')\big]$
$\Delta \leftarrow \max(\Delta, |v - V(s)|)$
until $\Delta < \theta$ (a small positive number)
Output $V \approx v_\pi$

Policy Evaluation Algorithm: SB example 4.1

0.0	-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	0.0



$R_t = -1$
on all transitions

What does this value become on third iteration?

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

$$= \sum_a \pi(a|s)[r + \gamma V^\pi(s')] \quad \text{(b/c deterministic)}$$

$$= \sum_a 0.25[-1 + \gamma V^\pi(s')]$$

$$= -\frac{2.75}{4} - \frac{3}{4} - \frac{3}{4} - \frac{1}{4}$$

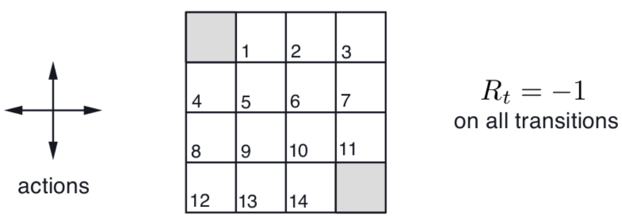
$$= -2.43$$

Input π, the policy to be evaluated
Initialize an array $V(s) = 0$, for all $s \in \mathcal{S}^+$
Repeat
    $\Delta \leftarrow 0$
    For each $s \in \mathcal{S}$:
        $v \leftarrow V(s)$
        $V(s) \leftarrow \sum_a \pi(a|s) \sum_{s',r} p(s', r|s, a)\big[r + \gamma V(s')\big]$
        $\Delta \leftarrow \max(\Delta, |v - V(s)|)$
until $\Delta < \theta$ (a small positive number)
Output $V \approx v_\pi$

Question

0.0	-14.	-20.	-22.
-14.	-18.	-20.	-20.
-20.	-20.	-18.	-14.
-22.	-20.	-14.	0.0

<u>Policy evaluation converges to these values</u>



Can you think of a simple interpretation of the values of these states when policy evaluation converges?

Input π, the policy to be evaluated
Initialize an array $V(s) = 0$, for all $s \in \mathcal{S}^+$
Repeat
    $\Delta \leftarrow 0$
    For each $s \in \mathcal{S}$:
        $v \leftarrow V(s)$
        $V(s) \leftarrow \sum_a \pi(a|s) \sum_{s',r} p(s',r|s,a)\big[r + \gamma V(s')\big]$
        $\Delta \leftarrow \max(\Delta, |v - V(s)|)$
until $\Delta < \theta$ (a small positive number)
Output $V \approx v_\pi$

Policy Improvement

Policy evaluation is great, but how do we use it to find better policies?

Policy Improvement

Policy evaluation is great, but how do we use it to find better policies?

Given: value function, $V^{\pi}(s)$, for a given policy π

Calculate: a new policy, π', that is at least as good as π

Policy Improvement

Policy evaluation is great, but how do we use it to find better policies?

Given: value function, $V^\pi(s)$, for a given policy π

Calculate: a new policy, π', that is at least as good as π

Policy improvement procedure:

1. calculate the action-value function, $V^\pi(s)$, for the latest policy, π

2. use Q to calculate a better policy: $\pi'(s) = \arg\max_a Q^\pi(s, a)$

Policy Improvement

Policy improvement procedure:

1. calculate the state-value function, $V^{\pi}(s)$ for the latest policy, π

2. use Q to calculate a better policy: $\pi'(s) = \arg\max_a Q^{\pi}(s,a)$

But, how do we calculate $Q^{\pi}(s,a)$ **from** $V^{\pi}(s)$ **?**

Policy Improvement

Value of being in state s , taking action a , and following policy π after that.

Value of being in state s, and following policy π after that.

But, how do we calculate $Q^{\pi}(s,a)$ **from** $V^{\pi}(s)$ **?**

Think-pair-share

Policy improvement procedure:

1. calculate the action-value function, $V^\pi(s)$, for the latest policy, π

2. use Q to calculate a better policy: $\pi'(s) = \arg\max_a Q^\pi(s,a)$

But, how do we calculate $Q^\pi(s,a)$ **from** $V^\pi(s)$ **?**

$$Q^\pi(s_t = s, a_t = a) =$$

Think-pair-share

Policy improvement procedure:

1. calculate the action-value function, $V^\pi(s)$, for the latest policy, π

2. use Q to calculate a better policy: $\pi'(s) = \arg\max_a Q^\pi(s,a)$

But, how do we calculate $Q^\pi(s,a)$ **from** $V^\pi(s)$ **?**

$$Q^\pi(s_t = s, a_t = a) = \boxed{}$$

<u>Hint: remember the Bellman eqn:</u>

$$V^\pi(s_t = s) = \sum_a \pi(a|s) \sum_{s',r} p(s',r|s,a)[r + \gamma V^\pi(s_{t+1} = s')]$$

Policy Improvement

Policy improvement procedure:

1. calculate the action-value function, $V^{\pi}(s)$, for the latest policy, π

2. use Q to calculate a better policy: $\pi'(s) = \arg\max_a Q^{\pi}(s,a)$

where $$Q^{\pi}(s,a) = \sum_{s',r} p(s',r|s,a)[r + \gamma V^{\pi}(s')]$$

Policy Improvement

Policy improvement procedure:

1. calculate the action-value function, $V^{\pi}(s)$, for the latest policy, π

2. use Q to calculate a better policy: $\pi'(s) = \arg\max_a Q^{\pi}(s,a)$

where $$Q^{\pi}(s,a) = \sum_{s',r} p(s',r|s,a)[r + \gamma V^{\pi}(s')]$$

Policy improvement theorem:

Let π and π' be arbitrary deterministic policies such that

$$Q^{\pi}(s, \pi'(s)) \geq V^{\pi}(s), \forall s \in \mathcal{S}$$ (SB, eqn 4.7)

Then $$V^{\pi'}(s) \geq V^{\pi}(s), \forall s \in \mathcal{S}$$ (SB, eqn 4.8)

Policy Improvement

Policy improvement procedure:

1. calculate the action-value function, $V^{\pi}(s)$, for the latest policy, π

2. use Q to calculate a

where $Q^{\pi}(s$

Policy π' is at least as good as π

Policy improvement theorem:

Let π and π' be arbitrary deterministic policies such that

$$Q^{\pi}(s, \pi'(s)) \geq V^{\pi}(s), \forall s \in \mathcal{S} \qquad \text{(SB, eqn 4.7)}$$

Then $V^{\pi'}(s) \geq V^{\pi}(s), \forall s \in \mathcal{S}$ (SB, eqn 4.8)

Policy Iteration

Policy iteration: combines policy evaluation and policy improvement

$$\pi_0 \xrightarrow{\text{E}} v_{\pi_0} \xrightarrow{\text{I}} \pi_1 \xrightarrow{\text{E}} v_{\pi_1} \xrightarrow{\text{I}} \pi_2 \xrightarrow{\text{E}} \cdots \xrightarrow{\text{I}} \pi_* \xrightarrow{\text{E}} v_*$$

Policy Iteration

Policy iteration: combines policy evaluation and policy improvement

$$\pi_0 \xrightarrow{\text{E}} v_{\pi_0} \xrightarrow{\text{I}} \pi_1 \xrightarrow{\text{E}} v_{\pi_1} \xrightarrow{\text{I}} \pi_2 \xrightarrow{\text{E}} \cdots \xrightarrow{\text{I}} \pi_* \xrightarrow{\text{E}} v_*$$

Arbitrary initial policy

Policy Iteration

Policy iteration: combines policy evaluation and policy improvement

$$\pi_0 \xrightarrow{\text{E}} v_{\pi_0} \xrightarrow{\text{I}} \pi_1 \xrightarrow{\text{E}} v_{\pi_1} \xrightarrow{\text{I}} \pi_2 \xrightarrow{\text{E}} \cdots \xrightarrow{\text{I}} \pi_* \xrightarrow{\text{E}} v_*$$

Policy evaluation

Policy Iteration

Policy iteration: combines policy evaluation and policy improvement

$$\pi_0 \xrightarrow{\text{E}} v_{\pi_0} \xrightarrow{\text{I}} \pi_1 \xrightarrow{\text{E}} v_{\pi_1} \xrightarrow{\text{I}} \pi_2 \xrightarrow{\text{E}} \cdots \xrightarrow{\text{I}} \pi_* \xrightarrow{\text{E}} v_*$$

Policy improvement

Policy Iteration

Policy iteration: combines policy evaluation and policy improvement

$$\pi_0 \xrightarrow{\text{E}} v_{\pi_0} \xrightarrow{\text{I}} \pi_1 \xrightarrow{\text{E}} v_{\pi_1} \xrightarrow{\text{I}} \pi_2 \xrightarrow{\text{E}} \cdots \xrightarrow{\text{I}} \pi_* \xrightarrow{\text{E}} v_*$$

New policy

Policy Iteration

Policy iteration: combines policy evaluation and policy improvement

$$\pi_0 \xrightarrow{\text{E}} v_{\pi_0} \xrightarrow{\text{I}} \pi_1 \xrightarrow{\text{E}} v_{\pi_1} \xrightarrow{\text{I}} \pi_2 \xrightarrow{\text{E}} \cdots \xrightarrow{\text{I}} \pi_* \xrightarrow{\text{E}} v_*$$

Policy evaluation

Policy Iteration

Policy iteration: combines policy evaluation and policy improvement

$$\pi_0 \xrightarrow{\text{E}} v_{\pi_0} \xrightarrow{\text{I}} \pi_1 \xrightarrow{\text{E}} v_{\pi_1} \xrightarrow{\text{I}} \pi_2 \xrightarrow{\text{E}} \cdots \xrightarrow{\text{I}} \pi_* \xrightarrow{\text{E}} v_*$$

Policy improvement

Policy Iteration

Policy iteration: combines policy evaluation and policy improvement

$$\pi_0 \xrightarrow{\text{E}} v_{\pi_0} \xrightarrow{\text{I}} \pi_1 \xrightarrow{\text{E}} v_{\pi_1} \xrightarrow{\text{I}} \pi_2 \xrightarrow{\text{E}} \cdots \xrightarrow{\text{I}} \pi_* \xrightarrow{\text{E}} v_*$$

New policy

Policy Iteration

<u>Policy iteration:</u> (SB pp 65)

1. Initialization
   $V(s) \in \mathbb{R}$ and $\pi(s) \in \mathcal{A}(s)$ arbitrarily for all $s \in \mathcal{S}$

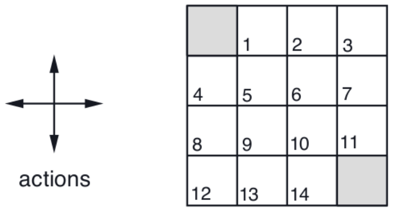
2. Policy Evaluation
   Repeat
   $\quad \Delta \leftarrow 0$
   $\quad$ For each $s \in \mathcal{S}$:
   $\quad\quad v \leftarrow V(s)$
   $\quad\quad V(s) \leftarrow \sum_{s',r} p(s', r | s, \pi(s)) \big[r + \gamma V(s') \big]$
   $\quad\quad \Delta \leftarrow \max(\Delta, |v - V(s)|)$
   until $\Delta < \theta$ (a small positive number)

3. Policy Improvement
   policy-stable $\leftarrow$ *true*
   For each $s \in \mathcal{S}$:
   $\quad$ *old-action* $\leftarrow \pi(s)$
   $\quad \pi(s) \leftarrow \text{argmax}_a \sum_{s',r} p(s', r | s, a) \big[r + \gamma V(s') \big]$
   $\quad$ If *old-action* $\neq \pi(s)$, then *policy-stable* $\leftarrow$ *false*
   If *policy-stable*, then stop and return $V \approx v_*$ and $\pi \approx \pi_*$; else go to 2

$$\pi(s) \leftarrow \arg\max_a Q^{\pi}(s, a)$$

Policy Iteration Example

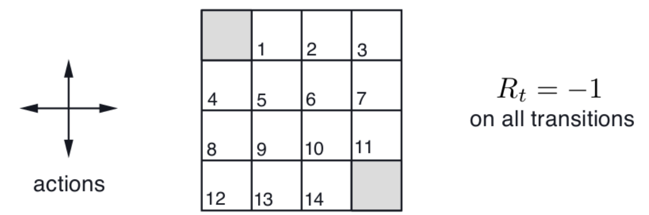
Recall grid world problem from earlier:



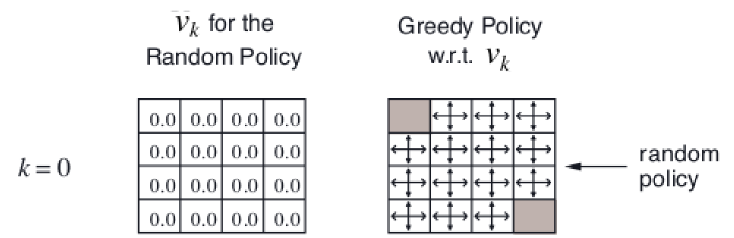
$R_t = -1$
on all transitions

Policy Iteration Example

Recall grid world problem from earlier:

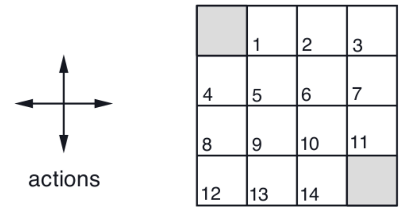


First iteration of PI:



Policy Iteration Example

Recall grid world problem from earlier:



$R_t = -1$
on all transitions

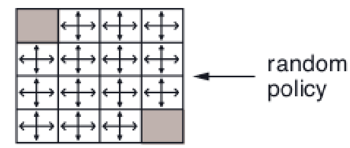
First iteration of PI:

v_k for the Random Policy

Greedy Policy w.r.t. v_k

$k = 0$

0.0	0.0	0.0	0.0
0.0	0.0	0.0	0.0
0.0	0.0	0.0	0.0
0.0	0.0	0.0	0.0



$k = 1$

0.0	-1.0	-1.0	-1.0
-1.0	-1.0	-1.0	-1.0
-1.0	-1.0	-1.0	-1.0
-1.0	-1.0	-1.0	0.0

$k = 2$

0.0	-1.7	-2.0	-2.0
-1.7	-2.0	-2.0	-2.0
-2.0	-2.0	-2.0	-1.7
-2.0	-2.0	-1.7	0.0

$k = 3$

0.0	-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	0.0

$k = 10$

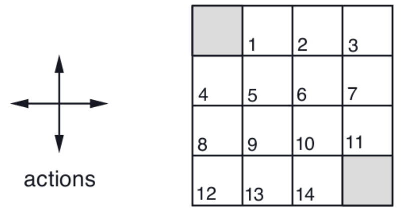
0.0	-6.1	-8.4	-9.0
-6.1	-7.7	-8.4	-8.4
-8.4	-8.4	-7.7	-6.1
-9.0	-8.4	-6.1	0.0

$k = \infty$

0.0	-14.	-20.	-22.
-14.	-18.	-20.	-20.
-20.	-20.	-18.	-14.
-22.	-20.	-14.	0.0

Policy Iteration Example

Recall grid world problem from earlier:



$R_t = -1$
on all transitions

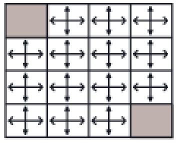
First iteration of PI:

v_k for the Random Policy

$k = 0$

0.0	0.0	0.0	0.0
0.0	0.0	0.0	0.0
0.0	0.0	0.0	0.0
0.0	0.0	0.0	0.0

Greedy Policy w.r.t. v_k



random policy

$k = 1$

0.0	-1.0	-1.0	-1.0
-1.0	-1.0	-1.0	-1.0
-1.0	-1.0	-1.0	-1.0
-1.0	-1.0	-1.0	0.0

$k = 2$

0.0	-1.7	-2.0	-2.0
-1.7	-2.0	-2.0	-2.0
-2.0	-2.0	-2.0	-1.7
-2.0	-2.0	-1.7	0.0

$k = 3$

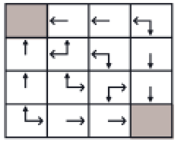
0.0	-2.4	-2.9	-3.0
-2.4	-2.9	-3.0	-2.9
-2.9	-3.0	-2.9	-2.4
-3.0	-2.9	-2.4	0.0

$k = 10$

0.0	-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	0.0

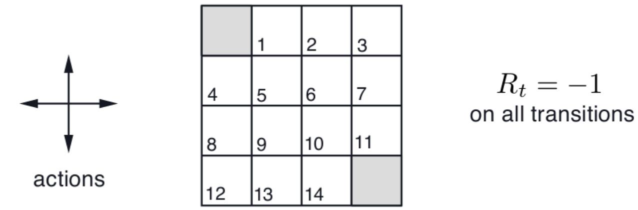
$k = \infty$

0.0	-14.	-20.	-22.
-14.	-18.	-20.	-20.
-20.	-20.	-18.	-14.
-22.	-20.	-14.	0.0



Policy Iteration Example

Recall grid world problem from earlier:



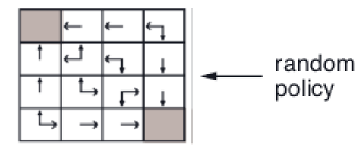
Second iteration of PI:

v_k for the Random Policy

$k = 0$

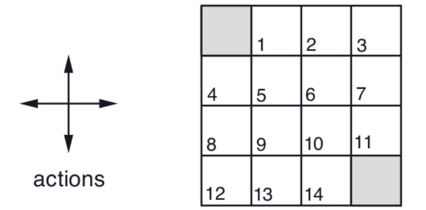
0.0	-14.	-20.	-22.
-14.	-18.	-20.	-20.
-20.	-20.	-18.	-14.
-22.	-20.	-14.	0.0

Greedy Policy w.r.t. v_k



Policy Iteration Example

Recall grid world problem from earlier:



$R_t = -1$
on all transitions

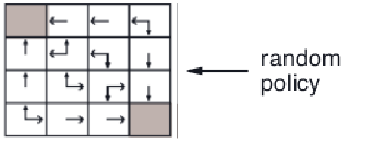
Second iteration of PI:

v_k for the Random Policy

$k = 0$

0.0	-14.	-20.	-22.
-14.	-18.	-20.	-20.
-20.	-20.	-18.	-14.
-22.	-20.	-14.	0.0

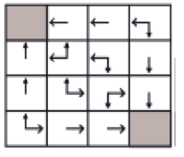
Greedy Policy w.r.t. v_k



•
•
•

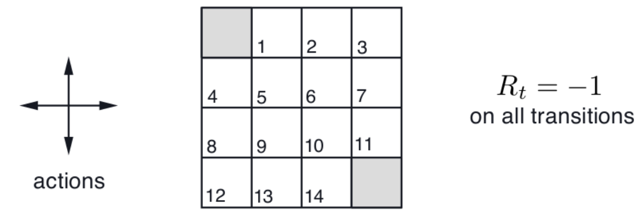
$k = \infty$

0.0	-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	0.0



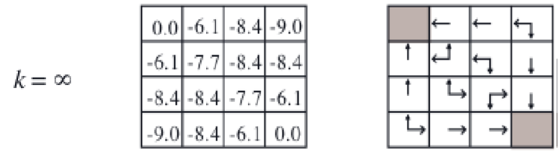
Policy Iteration Example

Recall grid world problem from earlier:



Second iteration of PI:

Notice that we don't need to run evaluation for many steps before the policy converges...



Think-pair-share

Exercise 4.5 How would policy iteration be defined for action values? Give a complete algorithm for computing q_*, analogous to that on page 80 for computing v_*. Please pay special attention to this exercise, because the ideas involved will be used throughout the rest of the book. □

1. Initialization
   $V(s) \in \mathbb{R}$ and $\pi(s) \in \mathcal{A}(s)$ arbitrarily for all $s \in \mathcal{S}$

2. Policy Evaluation
   Repeat
   - $\Delta \leftarrow 0$
   - For each $s \in \mathcal{S}$:
     - $v \leftarrow V(s)$
     - $V(s) \leftarrow \sum_{s',r} p(s', r \,|\, s, \pi(s)) \big[r + \gamma V(s') \big]$
     - $\Delta \leftarrow \max(\Delta, |v - V(s)|)$

   until $\Delta < \theta$ (a small positive number)

3. Policy Improvement
   policy-stable $\leftarrow$ *true*
   For each $s \in \mathcal{S}$:
   - *old-action* $\leftarrow \pi(s)$
   - $\pi(s) \leftarrow \operatorname{argmax}_a \sum_{s',r} p(s', r \,|\, s, a) \big[r + \gamma V(s') \big]$
   - If *old-action* $\neq \pi(s)$, then *policy-stable* $\leftarrow$ *false*

   If *policy-stable*, then stop and return $V \approx v_*$ and $\pi \approx \pi_*$; else go to 2

Stopping Policy Evaluation Early

Policy iteration: (SB pp 65)

1. Initialization
   $V(s) \in \mathbb{R}$ and $\pi(s) \in \mathcal{A}(s)$ arbitrarily for all $s \in \mathcal{S}$

2. Policy Evaluation
   Repeat
   - $\Delta \leftarrow 0$
   - For each $s \in \mathcal{S}$:
     - $v \leftarrow V(s)$
     - $V(s) \leftarrow \sum_{s',r} p(s', r \mid s, \pi(s)) \big[r + \gamma V(s')\big]$
     - $\Delta \leftarrow \max(\Delta, |v - V(s)|)$

   until $\Delta < \theta$ (a small positive number)

3. Policy Improvement
   policy-stable $\leftarrow$ *true*
   For each $s \in \mathcal{S}$:
   - *old-action* $\leftarrow \pi(s)$
   - $\pi(s) \leftarrow \arg\max_a \sum_{s',r} p(s', r \mid s, a) \big[r + \gamma V(s')\big]$
   - If *old-action* $\neq \pi(s)$, then *policy-stable* $\leftarrow$ *false*

   If *policy-stable*, then stop and return $V \approx v_*$ and $\pi \approx \pi_*$; else go to 2

Do we have to run policy evaluation until convergence, or can we stop early?

Stopping Policy Evaluation Early

Policy iteration: (SB pp 65)

1. Initialization
   $V(s) \in \mathbb{R}$ and $\pi(s) \in \mathcal{A}(s)$ arbitrarily for all $s \in \mathcal{S}$

2. Policy Evaluation
   Repeat
   $\Delta \leftarrow 0$
   For each $s \in \mathcal{S}$:
   $v \leftarrow V(s)$
   $V(s) \leftarrow \sum_{s',r} p(s',r|s,\pi(s))\big[r + \gamma V(s')\big]$
   $\Delta \leftarrow \max(\Delta, |v - V(s)|)$
   until $\Delta < \theta$ (a small positive number)

3. Policy Improvement
   policy-stable $\leftarrow$ *true*
   For each $s \in \mathcal{S}$:
   old-action $\leftarrow \pi(s)$
   $\pi(s) \leftarrow \mathrm{argmax}_a \sum_{s',r} p(s', r$ …
   If *old-action* $\neq \pi(s)$, then *policy-stable* $\leftarrow$ *false*
   If *policy-stable*, then stop and return $V \approx v_*$ and $\pi \approx \pi_*$; else go to 2

Do we have to run policy evaluation until convergence, or can we stop early?

We can stop early.
– how many iterations are needed?

Stopping Policy Evaluation Early

Policy iteration: (SB pp 65)

1. Initialization
   $V(s) \in \mathbb{R}$ and $\pi(s) \in \mathcal{A}(s)$ arbitrarily for all $s \in \mathcal{S}$

2. Policy Evaluation
   Repeat
   $\Delta \leftarrow 0$
   For each $s \in \mathcal{S}$:
   $v \leftarrow V(s)$
   $V(s) \leftarrow \sum_{s',r} p(s', r | s, \pi(s)) [r + \gamma V(s')]$
   $\Delta \leftarrow \max(\Delta, |v - V(s)|)$
   until $\Delta < \theta$ (a small positive number)

3. Policy Improvement
   policy-stable $\leftarrow$ *true*
   For each $s \in \mathcal{S}$:
   old-action $\leftarrow \pi(s)$
   $\pi(s) \leftarrow \mathrm{argmax}_a \sum_{s',r} p(s', r$
   If *old-action* $\neq \pi(s)$, then *policy-stable* $\leftarrow$ *false*
   If *policy-stable*, then stop and return $V \approx v_*$ and $\pi \approx \pi_*$; else go to 2

Do we have to run policy evaluation until convergence, or can we stop early?

We can stop early.
– how many iterations are needed?
– it turns out that even just one works.

Stopping Policy Evaluation Early

Policy iteration: (SB pp 65)

1. Initialization
   $V(s) \in \mathbb{R}$ and $\pi(s) \in \mathcal{A}(s)$ arbitrarily for all $s \in \mathcal{S}$

2. Policy Evaluation
   Repeat
   - $\Delta \leftarrow 0$
   - For each $s \in \mathcal{S}$:
     - $v \leftarrow V(s)$
     - $V(s) \leftarrow \sum_{s',r} p(s', r | s, \pi(s)) \big[r + \gamma V(s') \big]$
     - $\Delta \leftarrow \max(\Delta, |v - V(s)|)$

   until $\Delta < \theta$ (a small positive number)

3. Policy Improvement
   policy-stable $\leftarrow$ *true*
   For each $s \in \mathcal{S}$:
   - *old-action* $\leftarrow \pi(s)$
   - $\pi(s) \leftarrow \mathrm{argmax}_a \sum_{s',r} p(s', r$
   - If *old-action* $\neq \pi(s)$, then *policy-stable* $\leftarrow$ *false*

   If *policy-stable*, then stop and return $V \approx v_*$ and $\pi \approx \pi_*$; else go to 2

Do we have to run policy evaluation until convergence, or can we stop early?

We can stop early.
– how many iterations are needed?
– it turns out that even just one works.
– but any number is okay...

Stopping Policy Evaluation Early

Policy iteration: (SB pp 65)

1. Initialization
   $V(s) \in \mathbb{R}$ and $\pi(s) \in \mathcal{A}(s)$ arbitrarily for all $s \in \mathcal{S}$

2. Policy Evaluation
   Repeat
   $\Delta \leftarrow 0$
   For each $s \in \mathcal{S}$:
   $v \leftarrow V(s)$
   $V(s) \leftarrow \sum_{s',r} p(s',r|s,\pi(s))\big[r + \gamma V(s')\big]$
   $\Delta \leftarrow \max(\Delta, |v - V(s)|)$
   until $\Delta < \theta$ (a small positive number)

3. Policy Improvement
   policy-stable $\leftarrow$ *true*
   For each $s \in \mathcal{S}$:
   old-action $\leftarrow \pi(s)$
   $\pi(s) \leftarrow \text{argmax}_a \sum_{s',r} p(s', r$ ...
   If *old-action* $\neq \pi(s)$, then *policy-stable* $\leftarrow$ *false*
   If *policy-stable*, then stop and return $V \approx v_*$ and $\pi \approx \pi_*$; else go to ...

Do we have to run policy evaluation until convergence, or can we stop early?

We can stop early.
– how many iterations are needed?
– it turns out that even just one works.
– but any number is okay...

This is called value iteration

Value Iteration

Value iteration: (SB pp 67)

Initialize array V arbitrarily (e.g., $V(s) = 0$ for all $s \in \mathcal{S}^+$)

Repeat
    $\Delta \leftarrow 0$
    For each $s \in \mathcal{S}$:
        $v \leftarrow V(s)$
        $V(s) \leftarrow \max_a \sum_{s',r} p(s', r|s, a)\big[r + \gamma V(s')\big]$
        $\Delta \leftarrow \max(\Delta, |v - V(s)|)$
until $\Delta < \theta$ (a small positive number)

Output a deterministic policy, $\pi \approx \pi_*$, such that
    $\pi(s) = \arg\max_a \sum_{s',r} p(s', r|s, a)\big[r + \gamma V(s')\big]$

Value Iteration

Value iteration: (SB pp 67)

Initialize array V arbitrarily (e.g., $V(s) = 0$ for all $s \in \mathcal{S}^+$)

Policy improvement

One step of policy evaluation

Repeat

$\quad \Delta \leftarrow 0$

$\quad$ For each $s \in \mathcal{S}$:

$\quad\quad v \leftarrow V(s)$

$\quad\quad V(s) \leftarrow \max_a \sum_{s',r} p(s', r \mid s, a)\big[r + \gamma V(s')\big]$

$\quad\quad \Delta \leftarrow \max(\Delta, |v - V(s)|)$

until $\Delta < \theta$ (a small positive number)

Output a deterministic policy, $\pi \approx \pi_*$, such that

$\quad \pi(s) = \arg\max_a \sum_{s',r} p(s', r \mid s, a)\big[r + \gamma V(s')\big]$

Value Iteration

Value iteration: (SB pp 67)

Initialize array V arbitrarily (e.g., $V(s) = 0$ for all $s \in \mathcal{S}^+$)

Repeat

$\Delta \leftarrow 0$

For each $s \in \mathcal{S}$:

$v \leftarrow V(s)$

$V(s) \leftarrow \max_a \sum_{s',r} p(s', r \mid s, a)\big[r + \gamma V(s')\big]$

$\Delta \leftarrow \max(\Delta, |v - V(s)|)$

until $\Delta < \theta$ (a small positive number)

Output a deterministic policy, $\pi \approx \pi_*$, such that

$\pi(s) = \arg\max_a \sum_{s',r} p(s', r \mid s, a)\big[r + \gamma V(s')\big]$

Update rule based on Bellman Eqn

Another way to understand value iteration is to see that we are iteratively re-applying the Bellman Equation, used as an update rule.

Value Iteration

Another way to understand value iteration is to see that we are iteratively re-applying the Bellman Equation, used as an update rule.

1. $V_0(s) = \text{arbitrary}$
2. $V_1(s) = \max_a \sum_{s',r} p(s', r|s, a)[r + \gamma V_0(s')]$
3. $V_2(s) = \max_a \sum_{s',r} p(s', r|s, a)[r + \gamma V_1(s')]$
4. $V_3(s) = \max_a \sum_{s',r} p(s', r|s, a)[r + \gamma V_2(s')]$
5. $\vdots$

Value Iteration

Another way to understand value iteration is to see that we are iteratively re-applying the Bellman Equation, used as an update rule.

Initial value

1. $V_0(s) = \text{arbitrary}$
2. $V_1(s) = \max_a \sum_{s',r} p(s', r|s, a)[r + \gamma V_0(s')]$
3. $V_2(s) = \max_a \sum_{s',r} p(s', r|s, a)[r + \gamma V_1(s')]$
4. $V_3(s) = \max_a \sum_{s',r} p(s', r|s, a)[r + \gamma V_2(s')]$
5. $\vdots$

Value Iteration

Another way to understand value iteration is to see that we are iteratively re-applying the Bellman Equation, used as an update rule.

Value over time horizon == 1

1. $V_0(s) = \text{arbitrary}$
2. $V_1(s) = \max_a \sum_{s',r} p(s',r|s,a)[r + \gamma V_0(s')]$
3. $V_2(s) = \max_a \sum_{s',r} p(s',r|s,a)[r + \gamma V_1(s')]$
4. $V_3(s) = \max_a \sum_{s',r} p(s',r|s,a)[r + \gamma V_2(s')]$
5. $\vdots$

Value Iteration

Another way to understand value iteration is to see that we are iteratively re-applying the Bellman Equation, used as an update rule.

Value over time horizon == 2

1. $V_0(s) = \text{arbitrary}$
2. $V_1(s) = \max_a \sum_{s',r} p(s',r|s,a)[r + \gamma V_0(s')]$
3. $V_2(s) = \max_a \sum_{s',r} p(s',r|s,a)[r + \gamma V_1(s')]$
4. $V_3(s) = \max_a \sum_{s',r} p(s',r|s,a)[r + \gamma V_2(s')]$
5. $\vdots$

Value Iteration

Another way to understand value iteration is to see that we are iteratively re-applying the Bellman Equation, used as an update rule.

1. $V_0(s) = \text{arbitrary}$
2. $V_1(s) = \max_a \sum_{s',r} p(s',r|s,a)[r + \gamma V_0(s')]$
3. $V_2(s) = \max_a \sum_{s',r} p(s',r|s,a)[r + \gamma V_1(s')]$
4. $V_3(s) = \max_a \sum_{s',r} p(s',r|s,a)[r + \gamma V_2(s')]$
5. $\vdots$

Value over time horizon == 3

Value Iteration

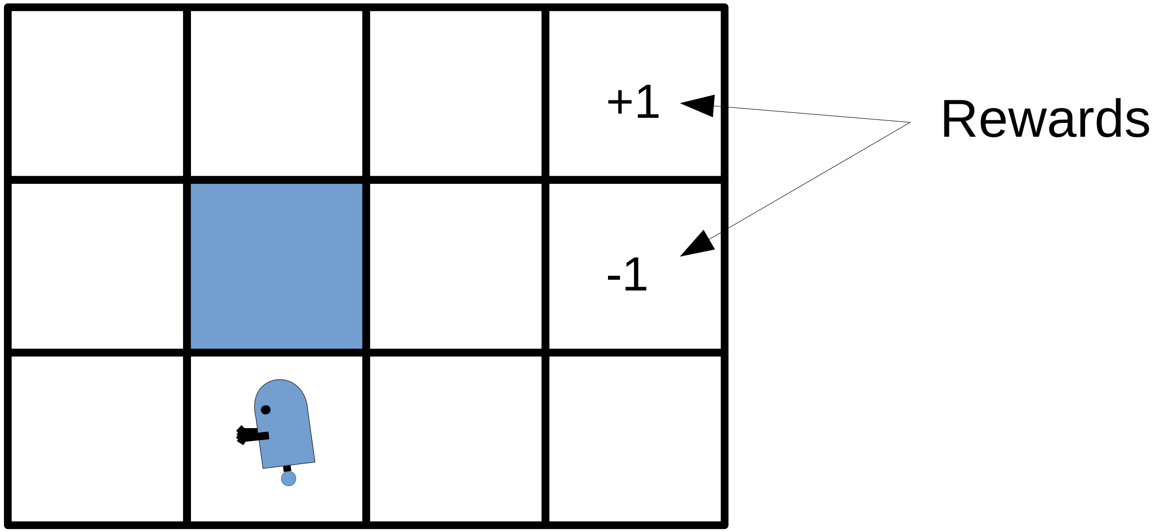
Another way to understand value iteration is to see that we are iteratively re-applying the Bellman Equation, used as an update rule.

Value over time horizon == 3

1. $V_0(s) = \text{arbitrary}$
2. $V_1(s) = \max_a \sum_{s',r} p(s',r|s,a)[r + \gamma V_0(s')]$
3. $V_2(s) = \max_a \sum_{s',r} p(s',r|s,a)[r + \gamma V_1(s')]$
4. $V_3(s) = \max_a \sum_{s',r} p(s',r|s,a)[r + \gamma V_2(s')]$
5. $\vdots$

Converges to optimal value function over infinite time horizon

Value iteration example



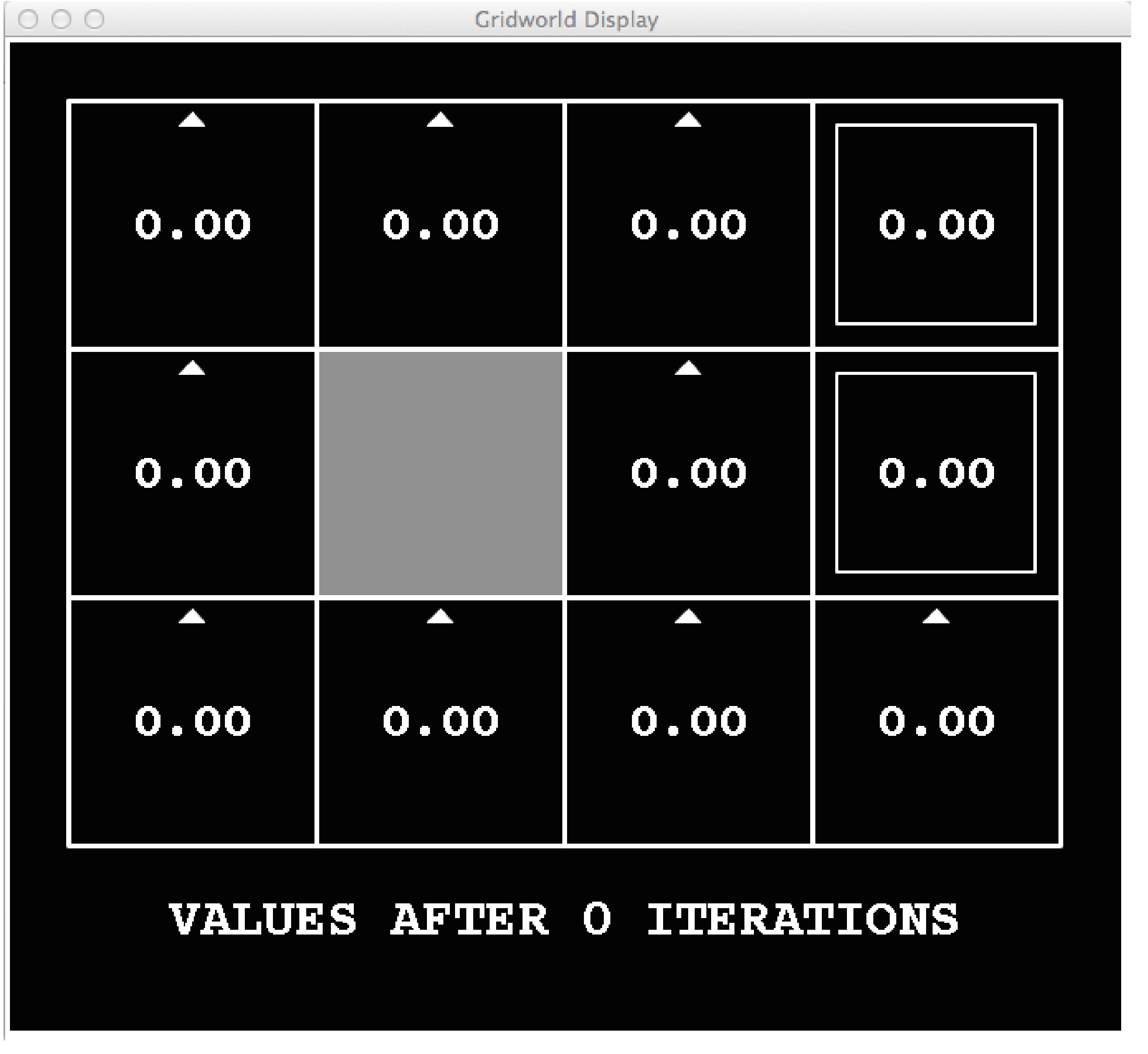
Noise = 0.2

Discount = 0.9

Living reward = 0

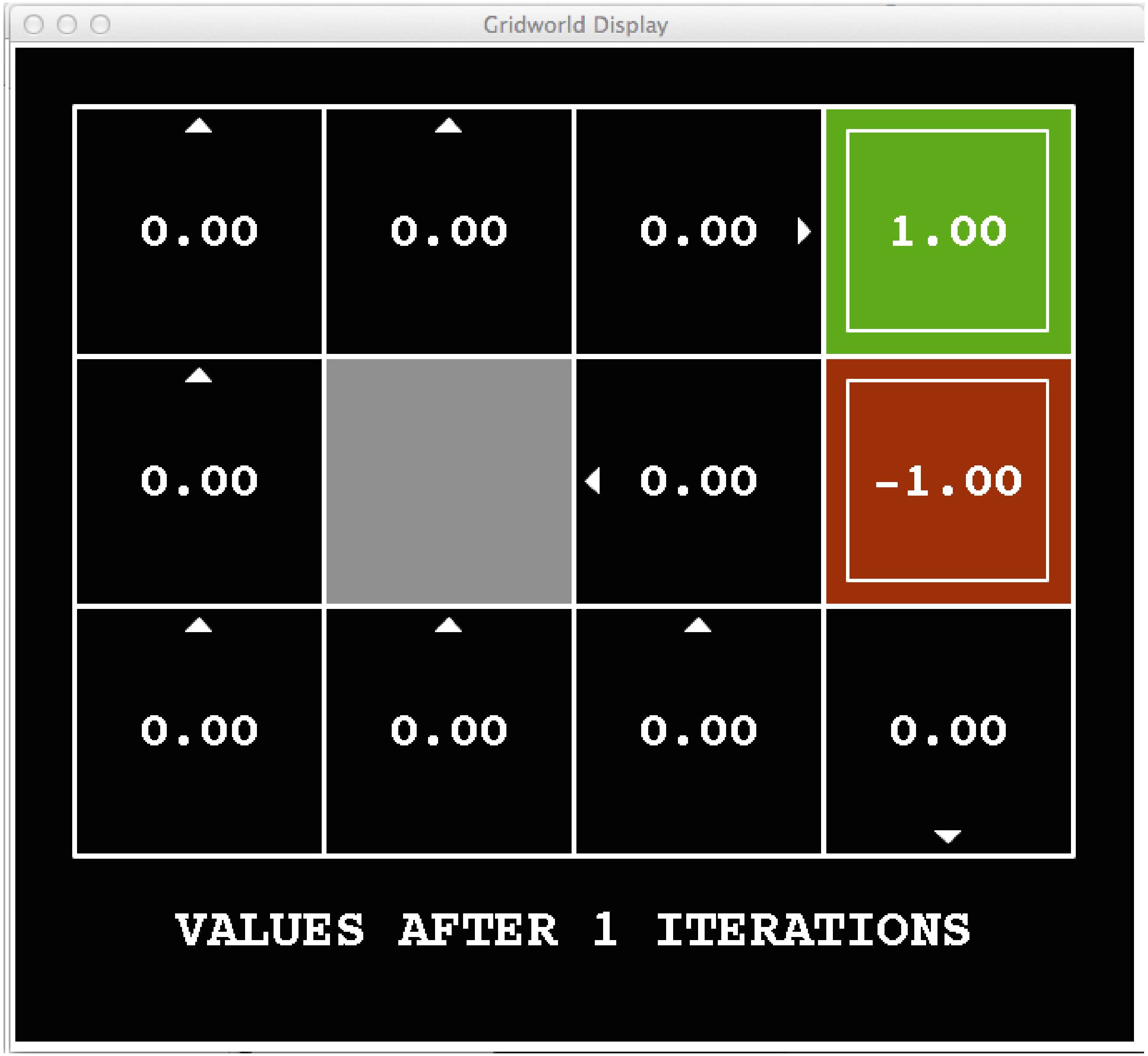
Actions: left, right, up, down
– take one action per time step
– actions are stochastic: only go in intended direction 80% of the time; 5% of time go in each of three other directions or stand still 5% of the time.

Value iteration example



Noise = 0.2
Discount = 0.9
Living reward = 0

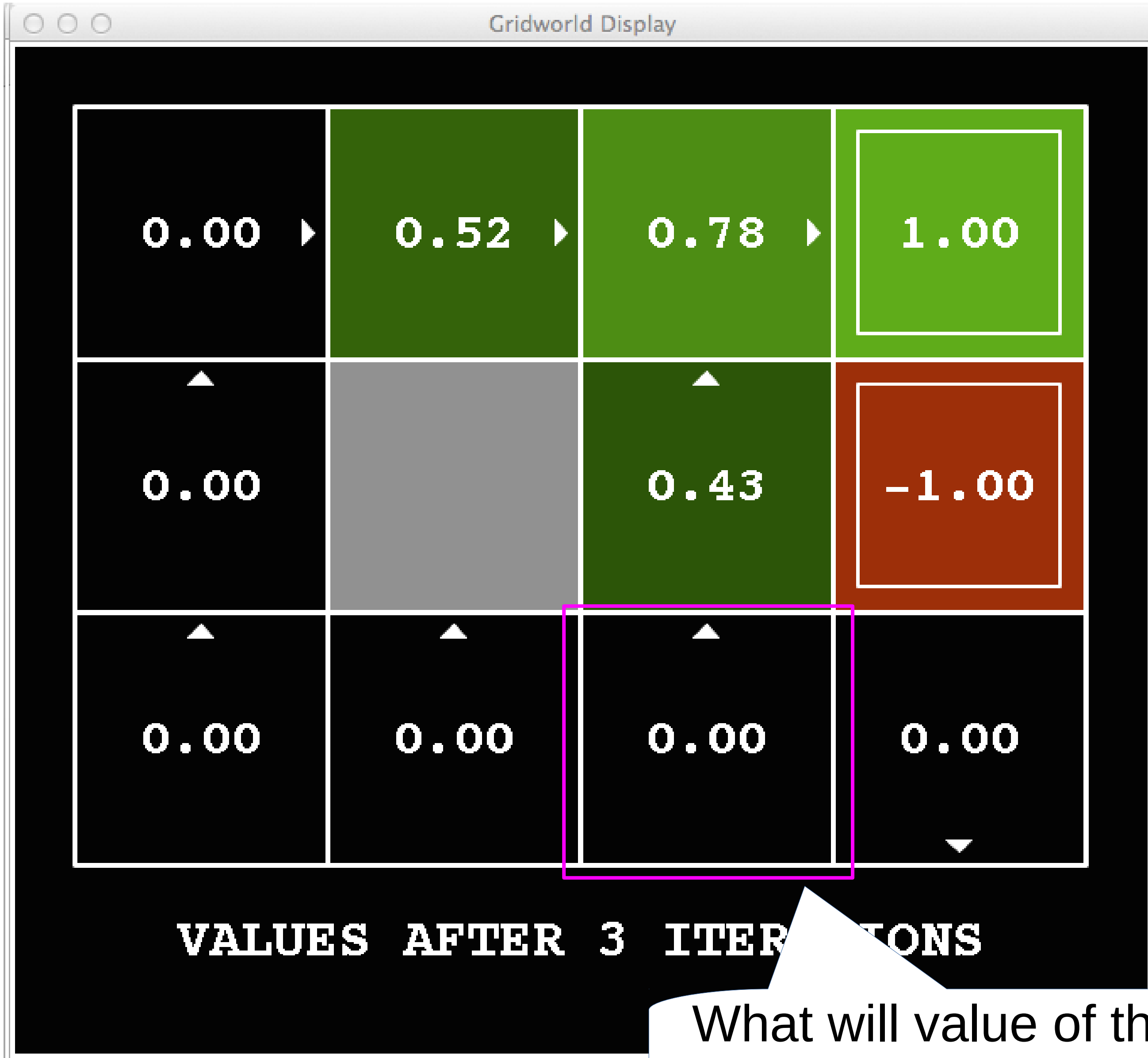
Value iteration example



Value iteration example

Value iteration example

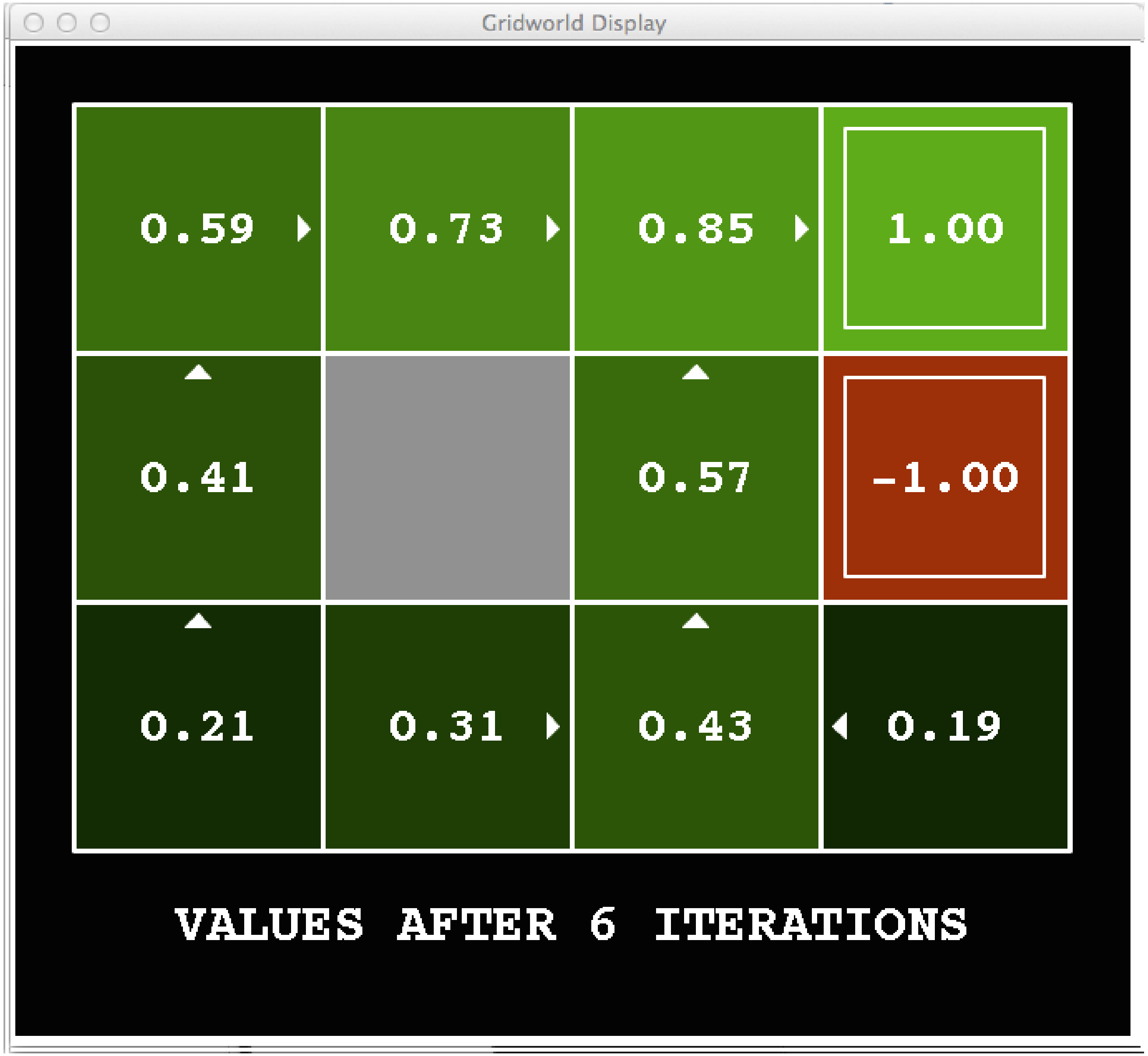
Think-pair-share



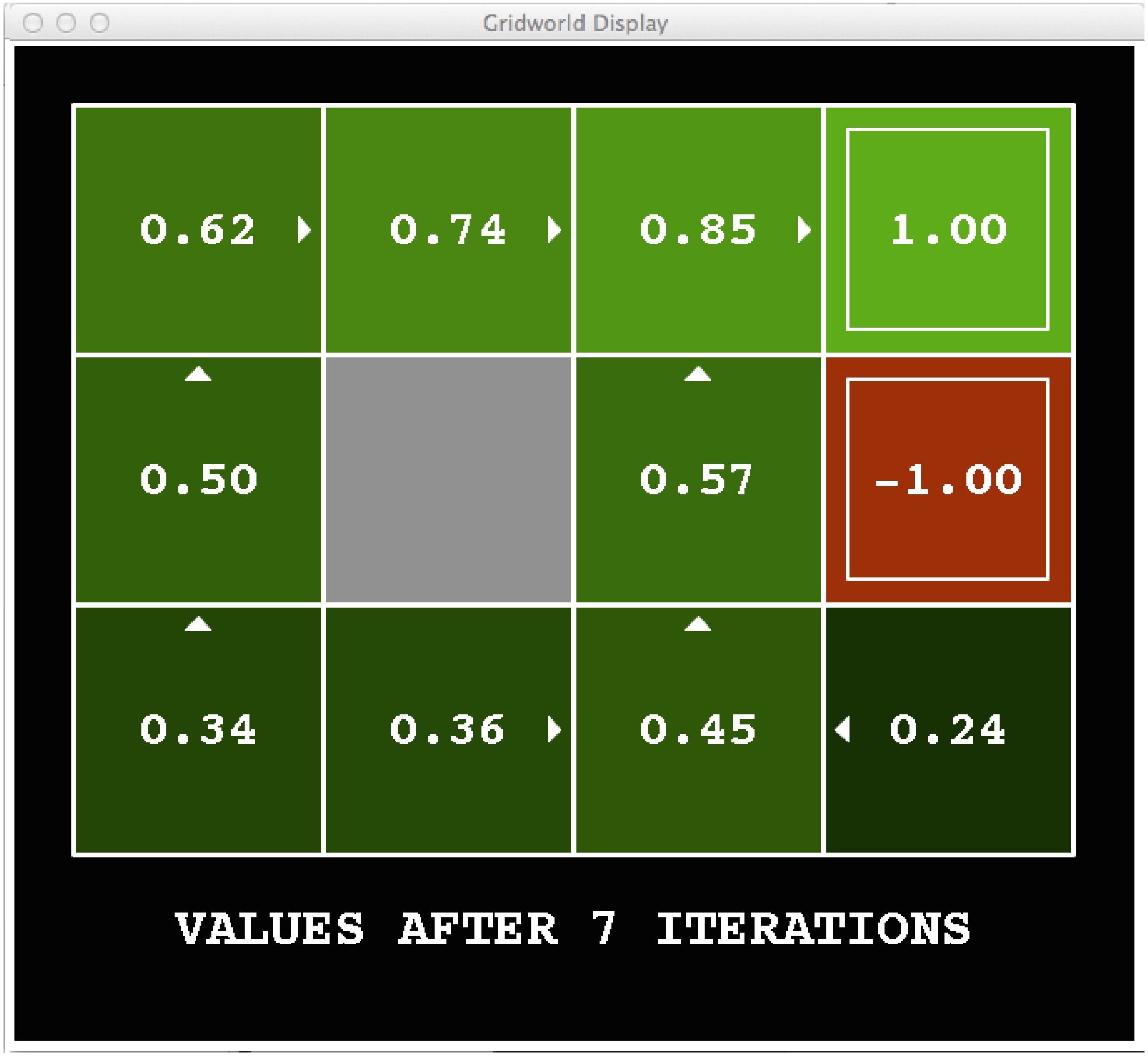
What will value of this state be on the next time step?

Value iteration example

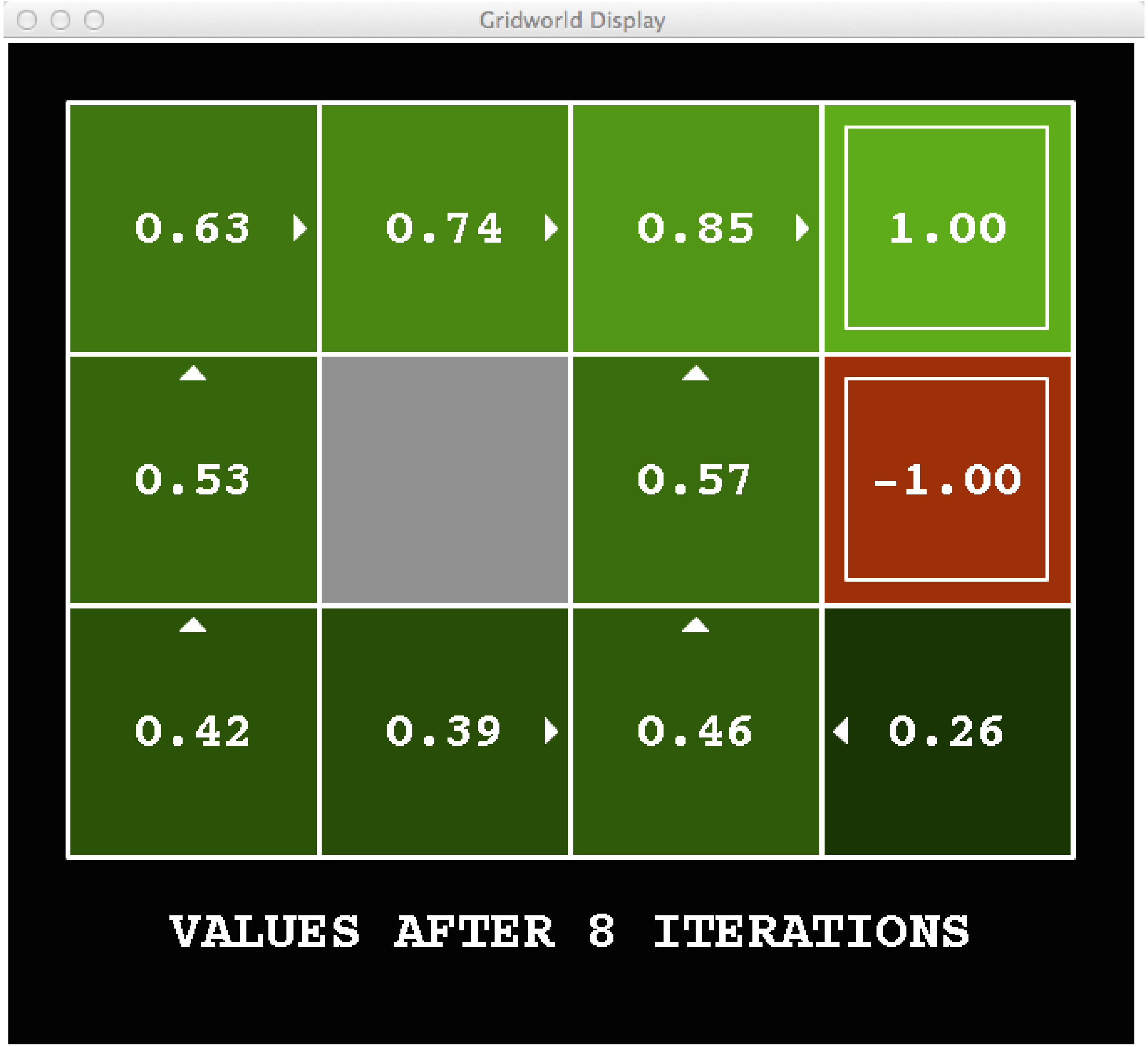
Value iteration example



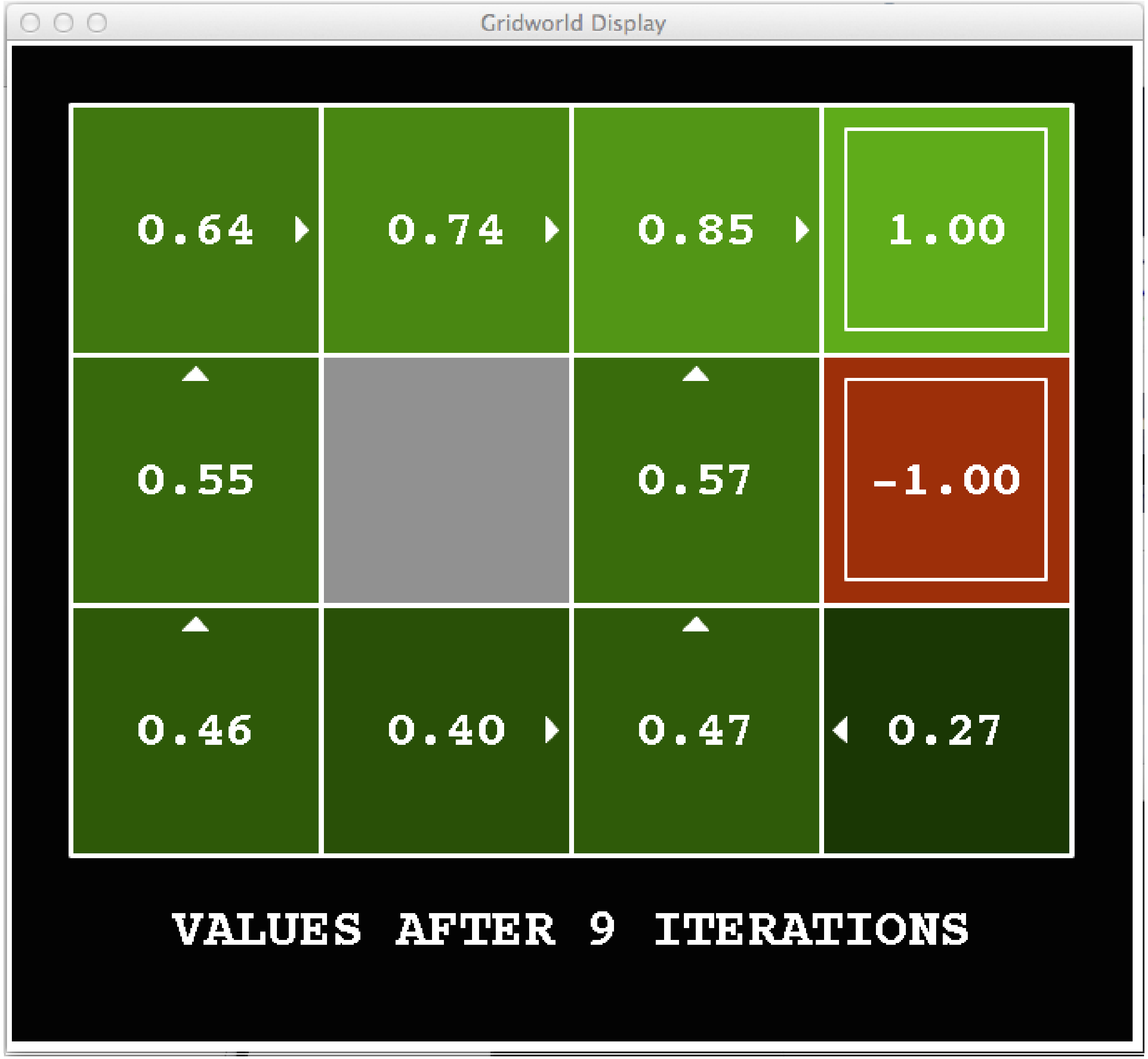
Value iteration example



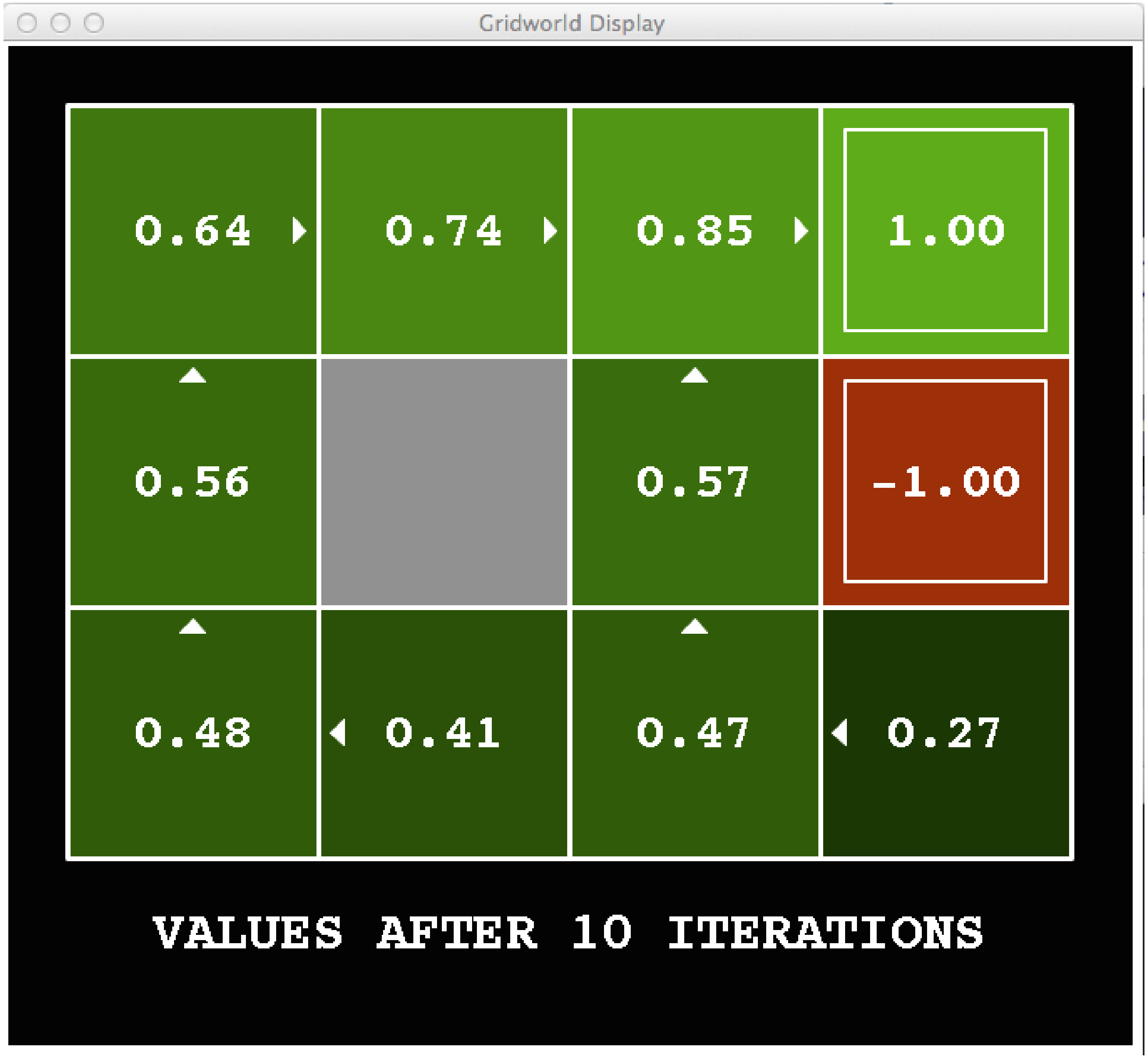
Value iteration example



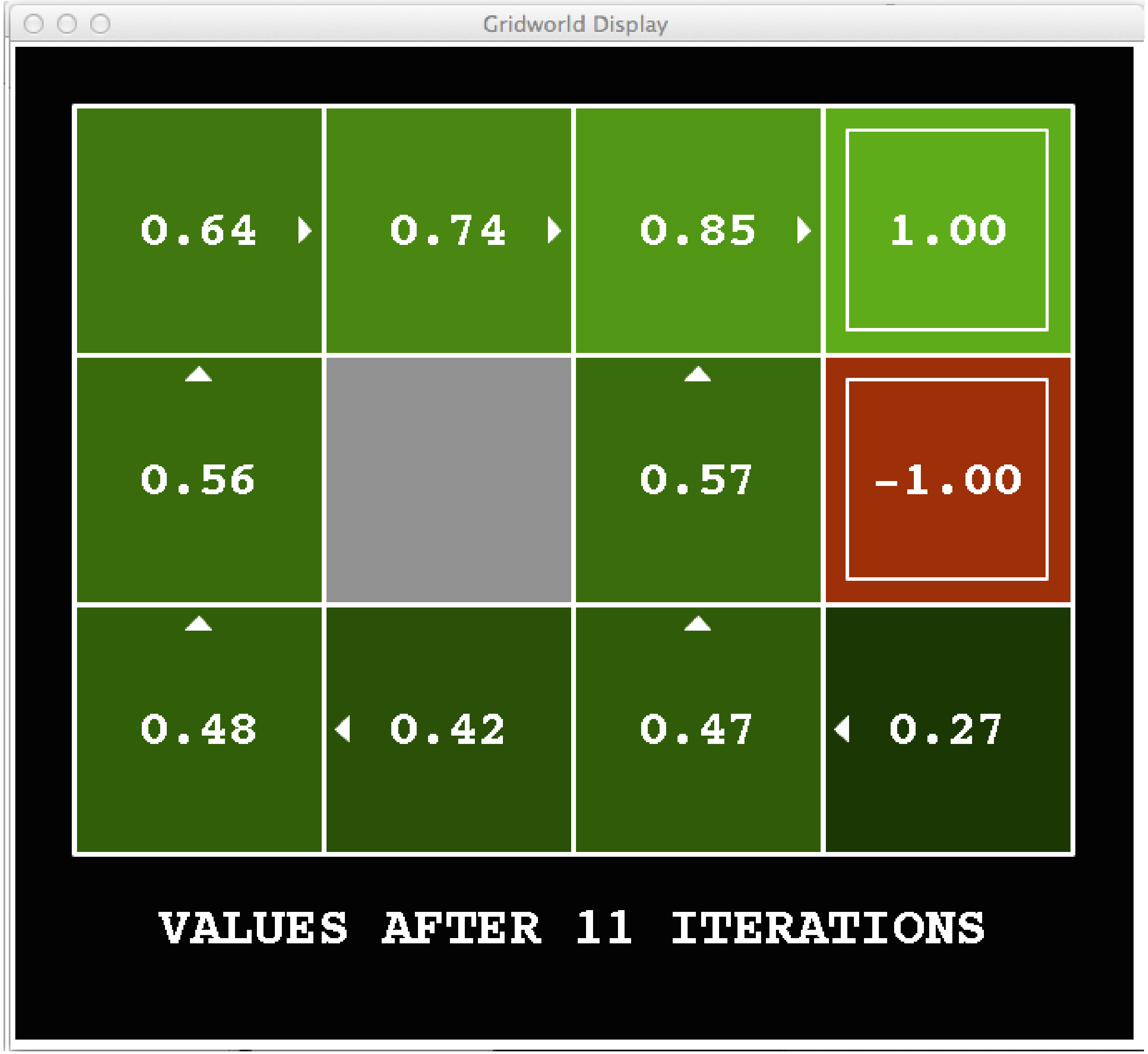
Value iteration example



Value iteration example

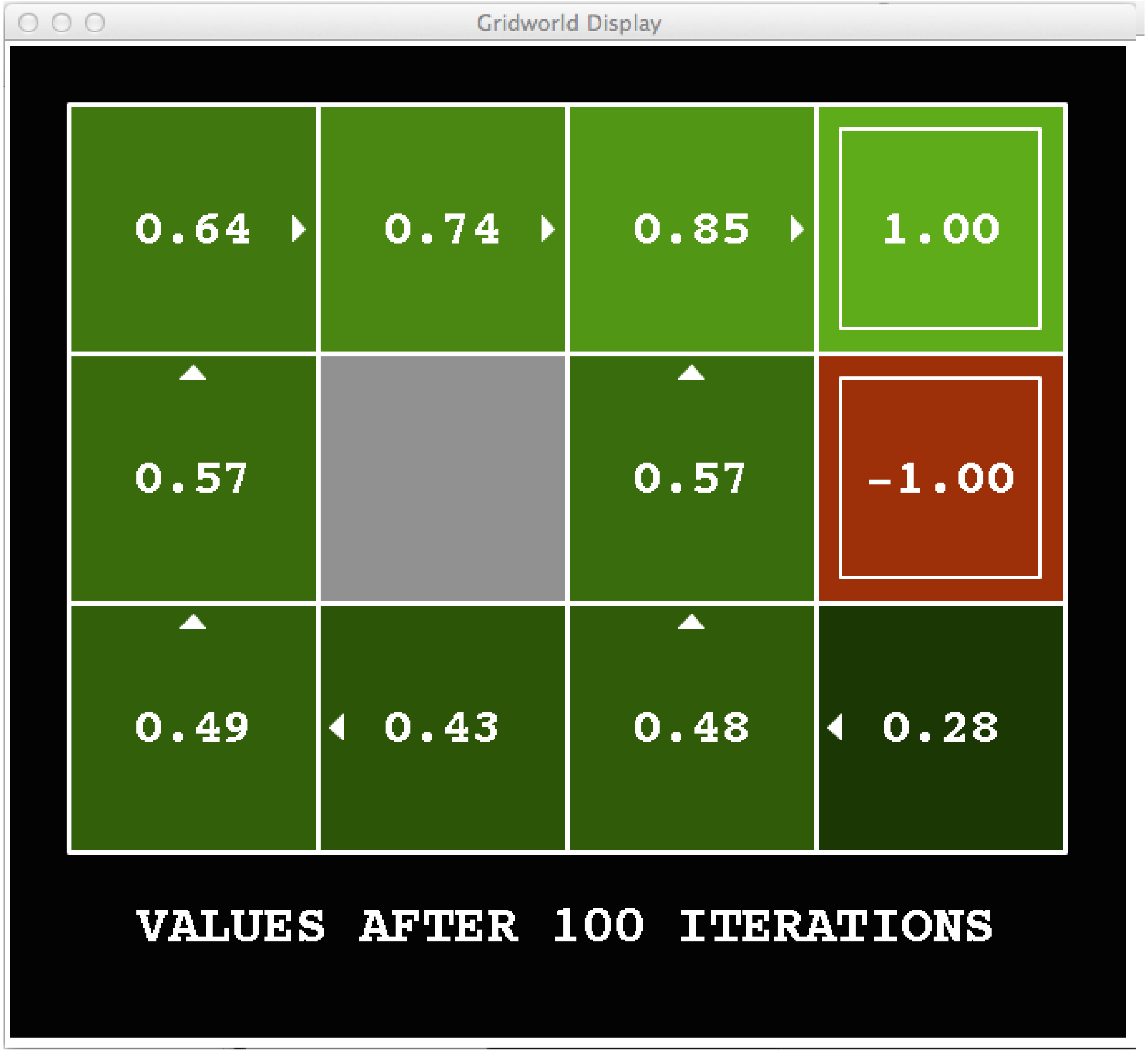


Value iteration example



Value iteration example

Value iteration example



Value Iteration Convergence

- How do we know the V_k vectors are going to converge?
- Case 1: If the tree has maximum depth M, then V_M holds the actual untruncated values
- Case 2: If the discount is less than 1
  - Sketch: For any state V_k and V_{k+1} can be viewed as depth k+1 expectimax results in nearly identical search trees
  - The last layer is at most all R_{MAX} and at least R_{MIN}
  - But everything is discounted by γ^k that far out
    - So V_k and V_{k+1} are at most $\gamma^k \max|R_{MAX} - R_{MIN}|$ different
  - So as k increases, the values converge

Value Iteration Optimality

At convergence, this property must hold

$$V(s) = \max_a \sum_{s'} P(s', r|s, a)[r + \gamma V(s')]$$

Question

At convergence, this property must hold (**why?**)

$$V(s) = \max_a \sum_{s'} P(s', r|s, a)[r + \gamma V(s')]$$

Value Iteration Optimality

At convergence, this property must hold (why?)

$$V(s) = \max_a \sum_{s'} P(s', r | s, a)[r + \gamma V(s')]$$

Initialize array V arbitrarily (e.g., $V(s) = 0$ for all $s \in \mathcal{S}^+$)

Repeat
 $\Delta \leftarrow 0$
 For each $s \in \mathcal{S}$:
  $v \leftarrow V(s)$
  $V(s) \leftarrow \max_a \sum_{s',r} p(s', r | s, a)\big[r + \gamma V(s')\big]$
  $\Delta \leftarrow \max(\Delta, |v - V(s)|)$
until $\Delta < \theta$ (a small positive number)

Output a deterministic policy, $\pi \approx \pi_*$, such that
 $\pi(s) = \arg\max_a \sum_{s',r} p(s', r | s, a)\big[r + \gamma V(s')\big]$

Value Iteration Optimality

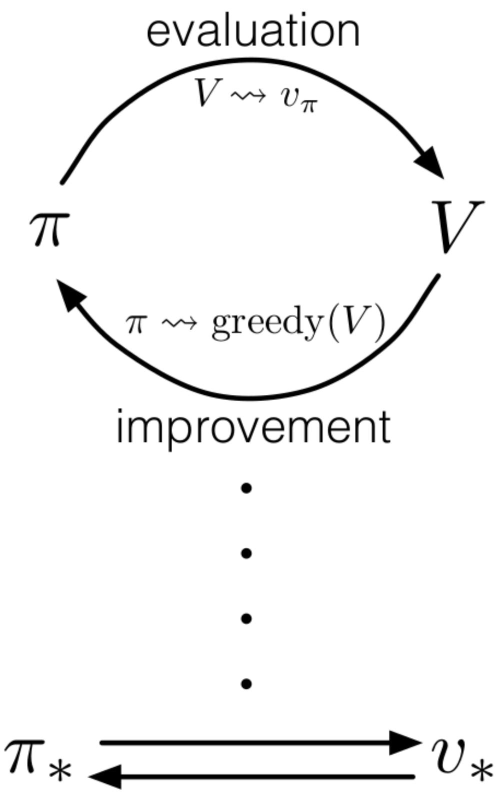
At convergence, this property must hold (why?)

$$V(s) = \max_a \sum_{s'} P(s', r|s, a)[r + \gamma V(s')]$$

What does this equation tell us about optimality of value iteration?

– we denote the *optimal* value function as: V^*

Generalized Policy Iteration



Policy iteration: (SB pp 65)

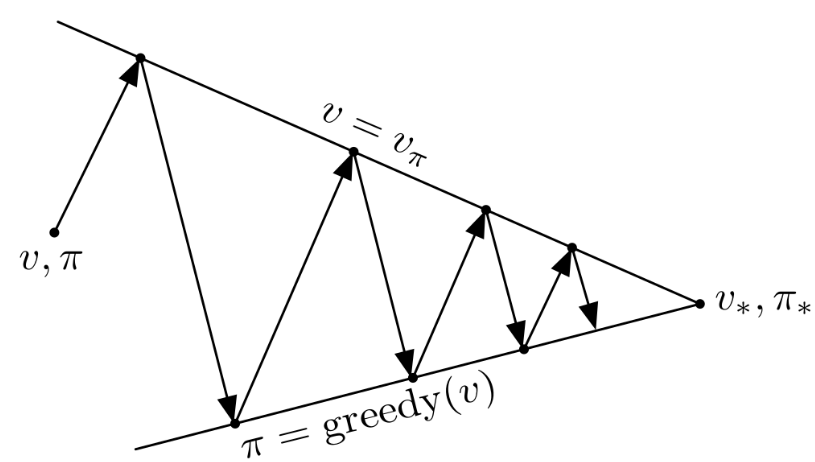
1. Initialization
   $V(s) \in \mathbb{R}$ and $\pi(s) \in \mathcal{A}(s)$ arbitrarily for all $s \in \mathcal{S}$

2. Policy Evaluation
   Repeat
   $\quad\Delta \leftarrow 0$
   $\quad$For each $s \in \mathcal{S}$:
   $\quad\quad v \leftarrow V(s)$
   $\quad\quad V(s) \leftarrow \sum_{s',r} p(s',r|s,\pi(s))\big[r + \gamma V(s')\big]$
   $\quad\quad \Delta \leftarrow \max(\Delta, |v - V(s)|)$
   until $\Delta < \theta$ (a small positive number)

3. Policy Improvement
   policy-stable $\leftarrow$ *true*
   For each $s \in \mathcal{S}$:
   $\quad$*old-action* $\leftarrow \pi(s)$
   $\quad\pi(s) \leftarrow \text{argmax}_a \sum_{s',r} p(s',r|s,a)\big[r + \gamma V(s')\big]$
   $\quad$If *old-action* $\neq \pi(s)$, then *policy-stable* $\leftarrow$ *false*
   If *policy-stable*, then stop and return $V \approx v_*$ and $\pi \approx \pi_*$; else go to 2

Some number of policy evaluation steps

Policy improvement



– can also udpate different parts of state space out of order
– generally, GPI converges as long as you can guarantee that every state is updated infinitely often

Computational efficiency of VI and PI

Policy iteration runs in time polynomial in the number of states and actions

– notice that tree search would need to consider an exponential number of paths through the state space (how many?)

– policy iteration finds the best policy in only polynomial time!

Summary

– policy evaluation

– policy improvement; policy improvement theorem

– policy iteration; convergence, optimality

– value iteration; convergence optimality