Can Machines Learn from Experience? (Part 2)

Optimization and Planning

Agenda

1. Last Week (Ch. 1, 3[1])
2. Bellman Optimality Principle (Ch. 3[1])
3. Planning (Ch. 4[1])
   a. Policy Evaluation
   b. Policy Improvement
   c. Policy Iteration

Reinforcement learning

- Formalization of learning through *interaction*
- What to do in order to maximize some numerical reward
    - Mapping states to actions, $\pi : S \to A$
- Characterized by the **problem** rather than a method(s)
    - Any method suited for RL problems can be considered an RL method

The Agent-Environment interface

- Agent
  - is learner and decision maker

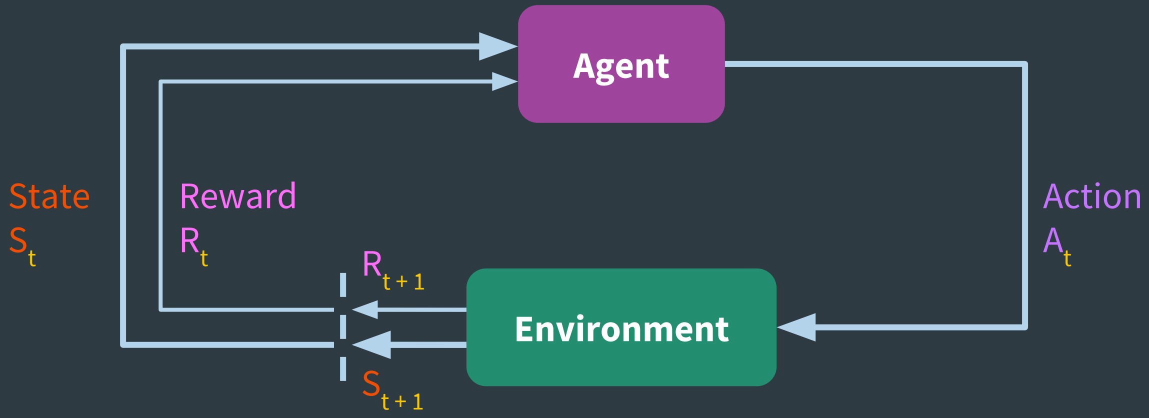
- Environment
  - intuitively: everything in agent *interacts with*
  - unintuitively: everything the agent is ***not***
  - gives rise to rewards (**R**), and has states (**S**)

- Task
  - single instance of the RL problem
  - complete definition of the environment and how rewards are determined

The A/E interface

- We deal with discrete timesteps
  - $t = 0, 1, \ldots, n \mid n \in \mathbb{Z}^+$
- At each time t the environment has some state (S_t)
  - $S_t \in \mathbf{S}$, where S is all possible states of the environment
- At each time t, given a state S_t, the agent takes an action (A_t)
  - $A_t \in \mathbf{A}(S_t)$, where $\mathbf{A}(S_t)$ is the set of actions available in S_t
- After taking an action A_t – the agent receives some reward in $t + 1$
  - $R_{t+1} \in \mathbf{R}$
- The goal of RL is to have an agent maximize reward over the long term

The A/E interface



$t = 0, 1, 2, 3, \ldots$

$S_t \in \mathbf{S}$

$A_t \in \mathbf{A}(S_t)$

$R_t \in \mathbf{R}$

Equations of interest

Policy	$\pi : S \rightarrow A$
Reward Function	$R : S \times A \rightarrow \text{value}$
Model	$T : S \times A \rightarrow p(S' \mid S, A)$

π: maps from perceived states in the environment to actions that should be taken

R: defines the goal in a given problem, what the agent seeks to maximize – short-term focus

T: mimics behavior of the environment; ultimately should allow for inference about how the environment will behave

The Markov Property

- If it is the case that, $p(s', r \mid s, a) = \Pr\{S_{t+1} = s', R_{t+1} = r \mid S_t = s, A_t = a\}$; then our task has the Markov Property
- This enables us to predict S_{t+1} and R_{t+1} given $S_t = s$ and $A_t = a$
- Can be shown just as powerful as having complete history
- Can be shown best policy for Markov is equal to best policy for complete histories

Aside: Discounting

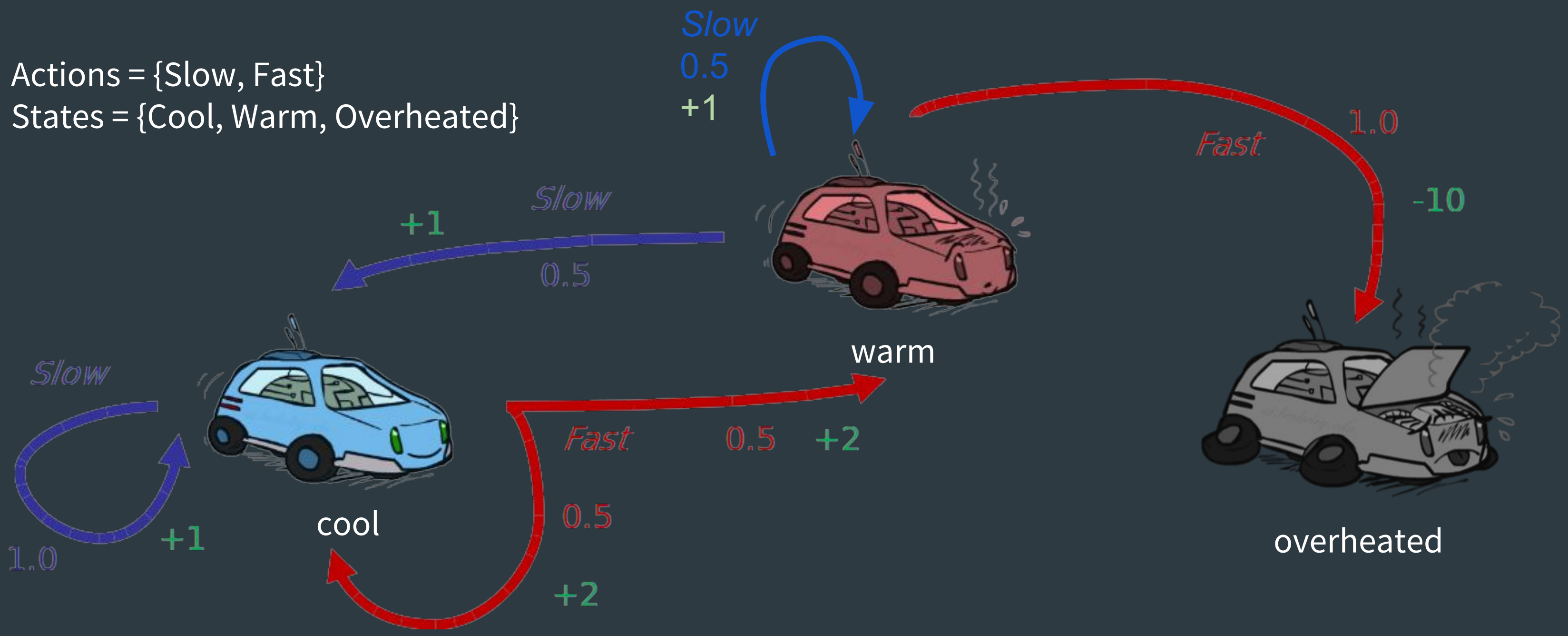
- The notion that rewards now are better than rewards later

- $0 \leq \gamma \leq 1$, the *discount factor* (how much do future rewards matter?)

- $G_t = R_{t+1} + R_{t+2} + R_{t+3} + \cdots + R_T$ becomes: $\Sigma_{k=\{0..\infty\}} (\gamma^k R_{t+k+1}) \mid k \to \infty$
- Now, if $\gamma < 1$, G_t has a finite value!
  (provided $\{R_k\}$ is bounded)

Markov Decision Process

Actions = {Slow, Fast}
States = {Cool, Warm, Overheated}



Graphics from [CS188 at Berkeley]

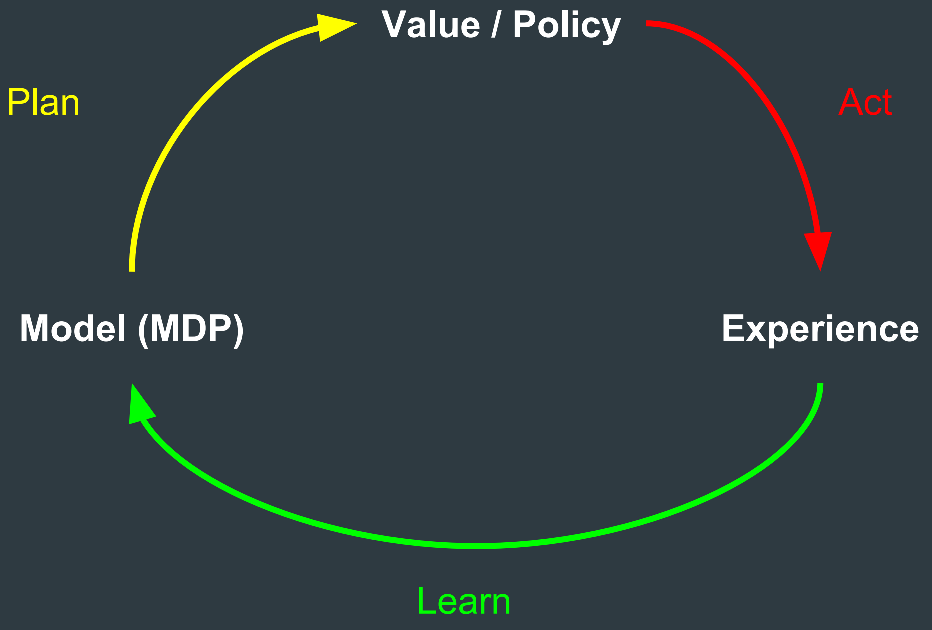
Goals and Rewards

The Reward Hypothesis:

> *That all of what we mean by goals and purposes can be well thought of as the maximization of the expected value of the cumulative sum of a received scalar signal (called reward).*

Basically, we can define a goal as maximizing some numerical value

Plan. Act. Learn.



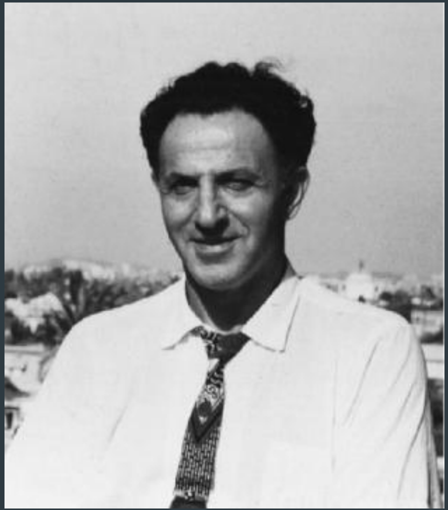
Agenda

1. Last Week (Ch. 1, 3[1])
2. Bellman Optimality Principle (Ch. 3[1])
3. Planning (Ch. 4[1])
    a. Policy Evaluation
    b. Policy Improvement
    c. Policy Iteration

Bellman Optimality Principle

"An optimal policy has the property that whatever the initial state and initial decision are, the remaining decisions must constitute an optimal policy with regard to the state resulting from the first decision"

-Richard Bellman



Bellman Optimality Principle

- Recall: $G = R_{t+1} + R_{t+2} + \cdot\cdot\cdot + R_T$ is the total reward
  - R_t = reward given at time t (denoted $R(s,a)$)
  - G becomes $\Sigma_{k=\{0..\infty\}} (\gamma^k R_{t+k+1}(s, a)) \mid k \to \infty$

- $Q^\pi(s, a)$ = Expected total reward of action a at state s w.r.t policy π

  $= E[G \mid S_t = s, A_t = a]$

  - A.K.A. the "Action-Value" function

Bellman Optimality Principle

- Focus on the action our policy maps to for a state s: $\pi(s) = a$
  - Pull this from $Q^{\pi}(s, a)$

- $V^{\pi}(s)$ = reward at state s w.r.t policy π

  $= E\,[G \mid S_t = s]$

  $= E\,[\Sigma_{k = \{0..\infty\}} (\gamma^k R_{t+k+1}(s, a)) \mid k \rightarrow \infty]$ converges to expected reward

  - A.K.A. the “State-Value” function

Bellman Optimality Principle

- Must be an optimal policy π^* for all possible policies
  - How do we find π^*?

Bellman Optimality Principle

- Must be an optimal policy π^* for all possible policies
  - How do we find π^*?

- π^* = π that yields highest value from its respective State-Value function

  $= \text{argmax}_{\pi} [V^{\pi}(s)]$, for any state s

Bellman Optimality Principle

- With π^*, we get our optimal equations

  For any state s :

Bellman Optimality Principle

- With π^*, we get our optimal equations

  For any state s :

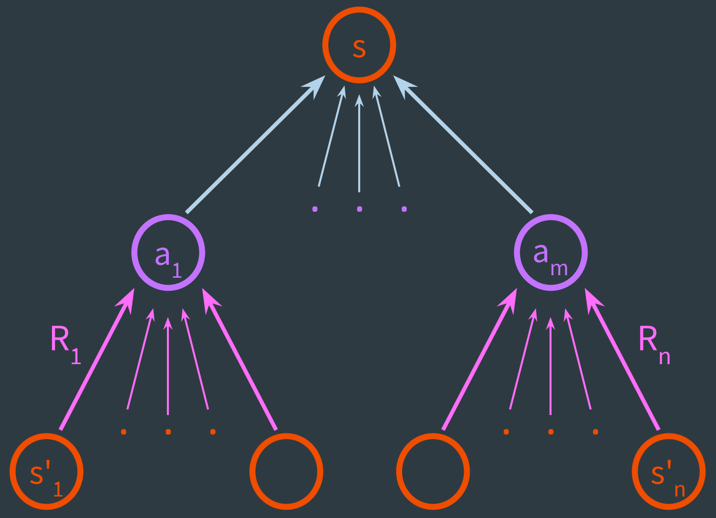
- $V^*(s) \quad = \max_a Q^*(s, a)$

Bellman Optimality Principle

- With π^*, we get our optimal equations

  For any state s :

- $V^*(s) \quad = \max_a Q^*(s, a)$
- $Q^*(s, a) \quad = R(s, a) + \Sigma_{s'} [T(s, a, s') \times V^*(s')]$



Agenda

1. Last Week (Ch. 1, 3[1])
2. Bellman Optimality Principle (Ch. 3[1])
3. Planning (Ch. 4[1])
   a. **Policy Evaluation**
   b. Policy Improvement
   c. Policy Iteration

Policy Evaluation

How do we evaluate a given policy π?

Policy Evaluation

How do we evaluate a given policy π?

Iterate the State-Value function:

Policy Evaluation

How do we evaluate a given policy π?

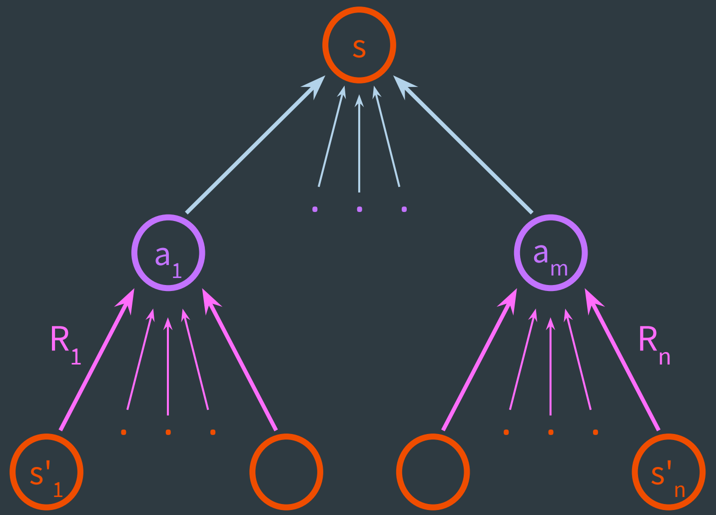
Iterate the State-Value function:

$$V_1 \rightarrow V_2 \rightarrow \ldots \rightarrow V^{\pi}$$

Policy Evaluation

- Synchronous full backup
  - Update $V_{t+1}(s)$ from $V_t(s')$
  - s’ successor of s
- Want to show convergence:

$$V_{t+1}(s) = \Sigma_a \pi(a \mid s) \times Q^{\pi}(s, a) = V^{\pi}(s)$$



Policy Evaluation in Grid-World

- Consider the following:
  - $\gamma = 1$
  - r = -1 (all transitions)
  - **A** = {left, right, up, down}
  - **S** = {1, … , 14}
  - one terminal state (shown twice)

$$V_{t+1}(s) = \Sigma_a \pi(a \mid s) \times Q^{\pi}(s, a) = V^{\pi}(s)$$

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

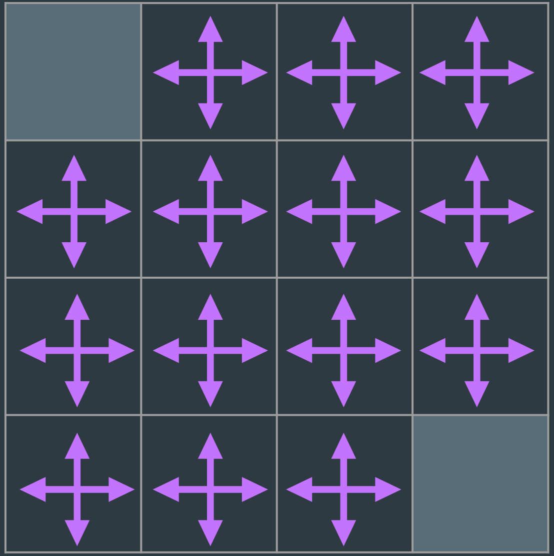
Policy Evaluation in Grid-World

v_0 for random policy

0.0	$V_0(1)$	$V_0(2)$	$V_0(3)$
$V_0(4)$	$V_0(5)$	$V_0(6)$	$V_0(7)$
$V_0(8)$	$V_0(9)$	$V_0(10)$	$V_0(11)$
$V_0(12)$	$V_0(13)$	$V_0(14)$	0.0

$k = 0$

Greedy policy



$$V_{t+1}(s) = \Sigma_a \pi(a \mid s) \times [R(s', a) + \Sigma_{s'} T(s, a, s') \times V_t(s')]$$

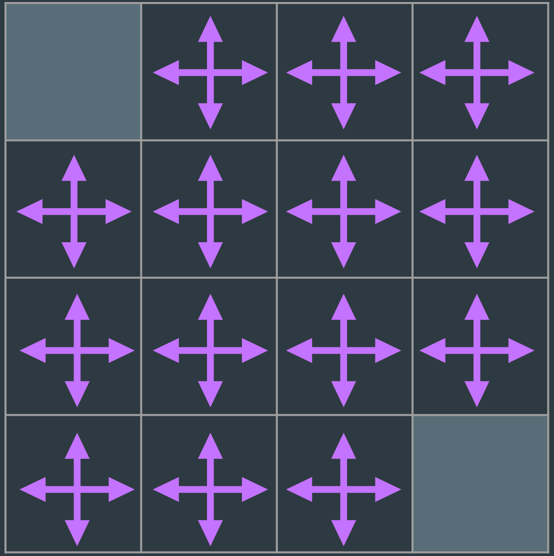
Policy Evaluation in Grid-World

v_0 for random policy

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

$k = 0$

Greedy policy



$$V_{t+1}(s) = \Sigma_a \pi(a \mid s) \times [R(s', a) + \Sigma_{s'} T(s, a, s') \times V_t(s')]$$

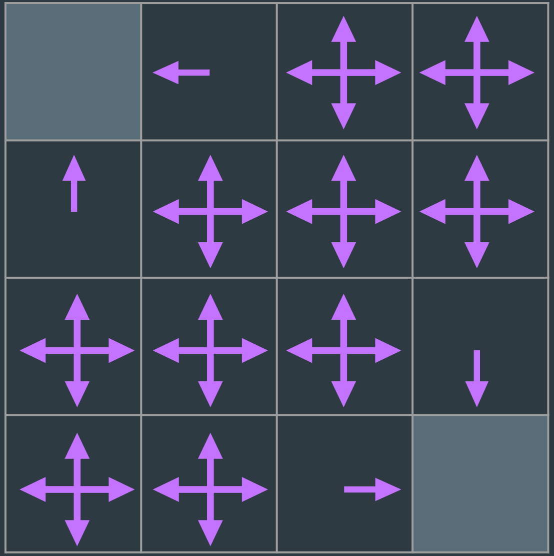
Policy Evaluation in Grid-World

v_1 for random policy

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

$k = 1$

Greedy policy



$$V_1(s) = \Sigma_a \pi(a \mid s) \times [-1.0 + \Sigma_{s'} 0.25 \times 0.0]$$

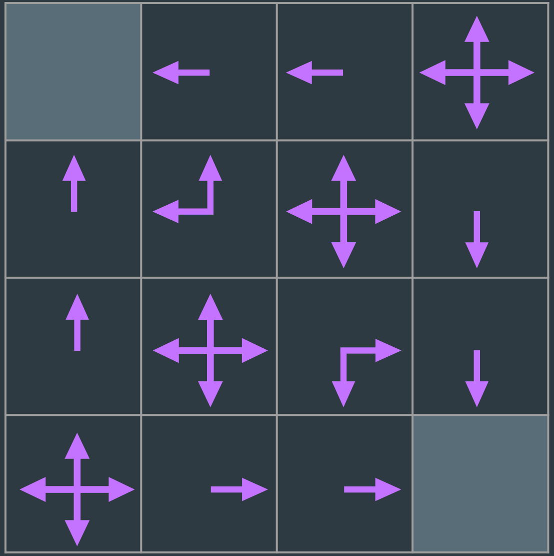
Policy Evaluation in Grid-World

v_2 for random policy

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

$k = 2$

Greedy policy



$$V_{t+1}(s) = \Sigma_a 0.25 \times [R(s, a) + \Sigma_{s'} 0.25 \times -1.0]$$

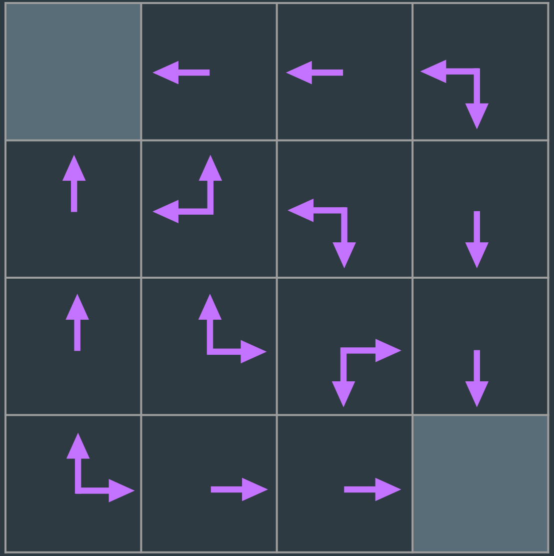
Policy Evaluation in Grid-World

v_3 for random policy

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

$k = 3$

Greedy policy



$$V_{t+1}(s) = \Sigma_a 0.25 \times [R(s, a) + \Sigma_{s'} 0.25 \times V_t(s')]$$

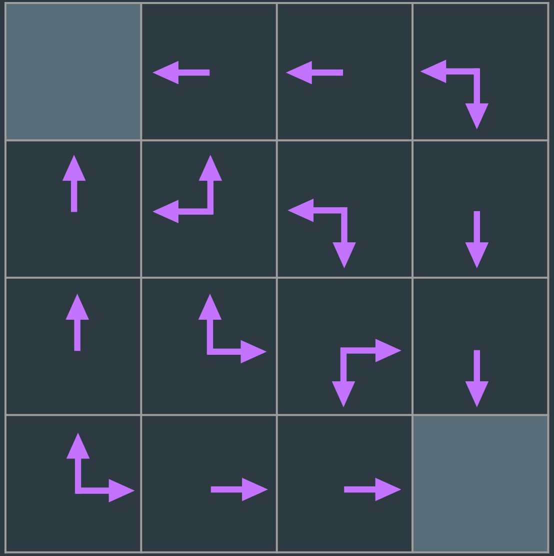
Policy Evaluation in Grid-World

v_{10} for random policy

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

$k = 10$

Greedy policy



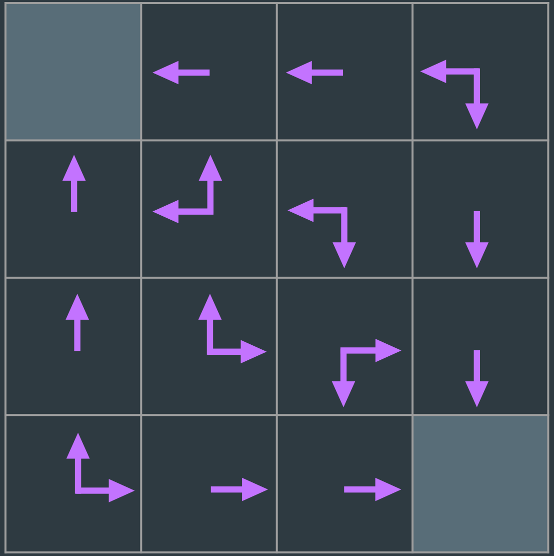
Policy Evaluation in Grid-World

v_∞ for random policy

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

$k = \infty$

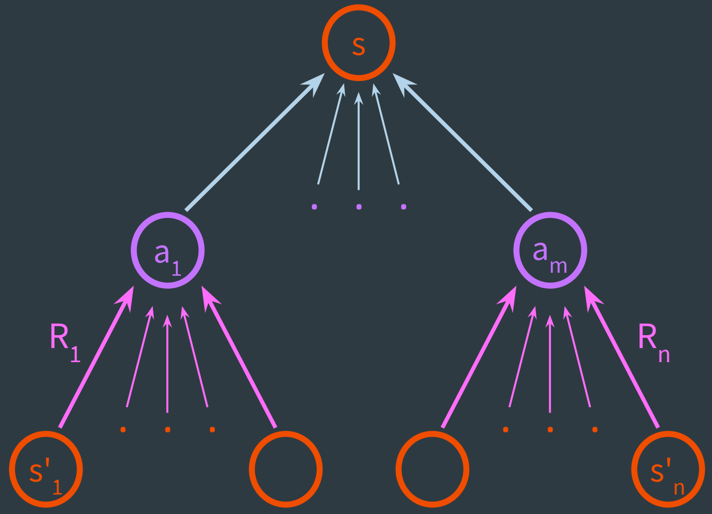
Greedy policy



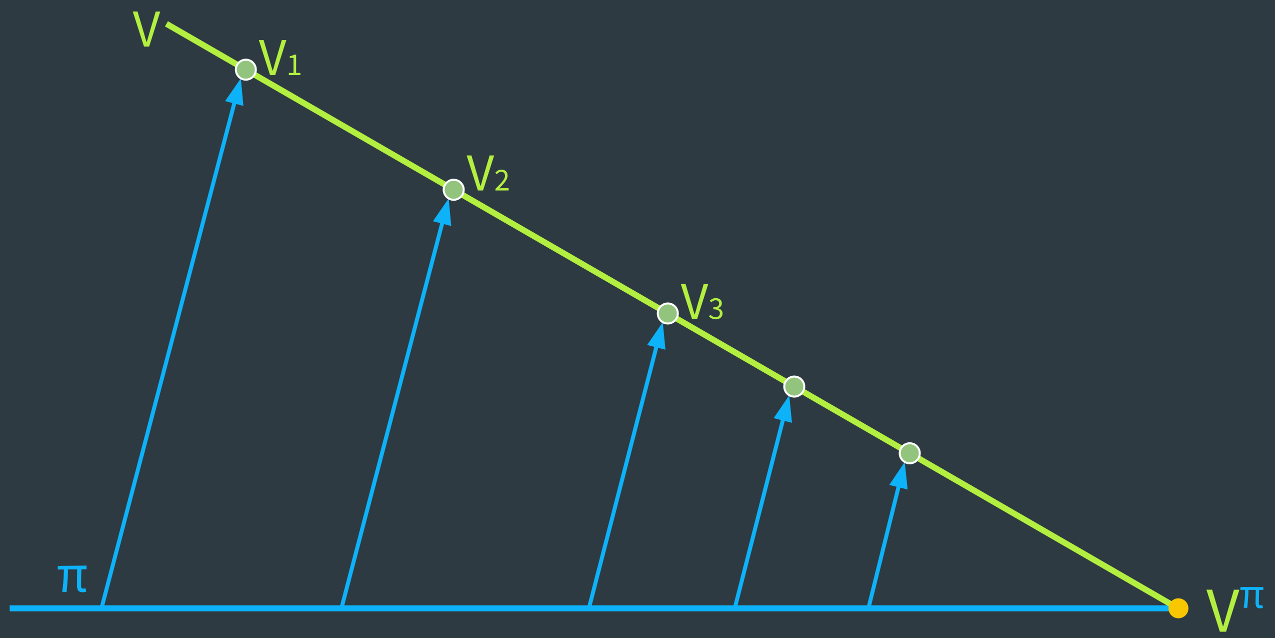
$$V_{t+1}(s) = \Sigma_a 0.25 \times [R(s, a) + \Sigma_{s'} 0.25 \times V_t(s')] = V^\pi(s)$$

Policy Evaluation Algorithm

```
policyEval(π):[

  threshold = (small positive #)
  delta = 0

  while ( delta >= threshold ):

    for s in S:
      v = V[s]

      for a in A:
        Q[s][a] = sum(S', p(s', s, a) * (R(s', s, a) + V[s']))

      V[s] = sum(A, π[s][a] * Q[s])
     delta = max(delta, abs(v - V[s]))

  return V  # approximate Vπ
```



Policy Evaluation Visualization



Agenda

1. Last Week (Ch. 1, 3[1])
2. Bellman Optimality Principle (Ch. 3[1])
3. Planning (Ch. 4[1])
    a. Policy Evaluation
    b. **Policy Improvement**
    c. Policy Iteration

Policy Improvement

How do we know a policy π is optimal?

Policy Improvement

How do we know a policy π is optimal?

Check for a better policy π'

Policy Improvement

Consider selecting *a* in *s* following current policy π such that:

For $\pi'(s) = a = \text{argmax}_a \, Q^{\pi}(s,a)$

If $Q^{\pi}(s,\pi'(s)) = \qquad \geq \qquad = V^{\pi}(s)$

Then this new policy π' must be good as, or better than, π.

So $V^{\pi'}(s) \geq V^{\pi}(s)$

Policy Improvement

Consider selecting *a* in *s* following current policy π such that:

For $\pi'(s) = a = \text{argmax}_a \, Q^{\pi}(s,a)$

If $V^{\pi'}(s) = \max_a \, Q^{\pi}(s,\pi'(s)) \geq Q^{\pi}(s,\pi'(s)) = V^{\pi}(s)$

Then this new policy π' must be good as, or better than, π.

So $V^{\pi'}(s) \geq V^{\pi}(s)$

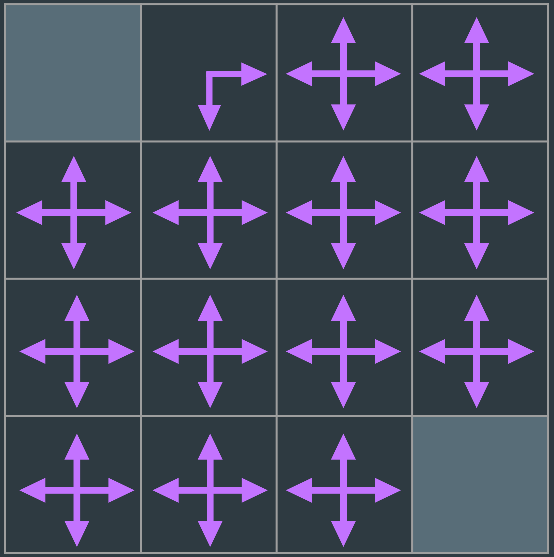
Policy Improvement in Grid-World

V_0 for random policy

	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	

$k = 0$

Stupid policy



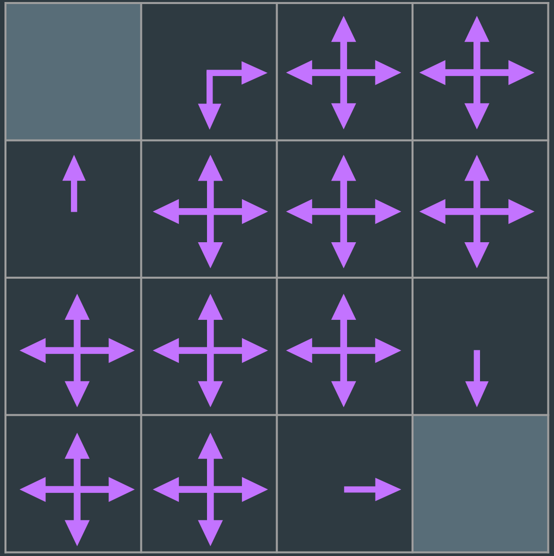
Policy Improvement in Grid-World

V_1 for random policy

	-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	

$k = 1$

Stupid policy



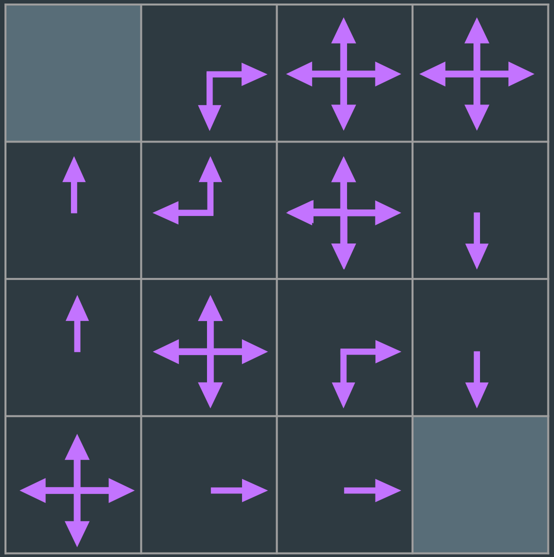
Policy Improvement in Grid-World

V_2 for random policy

	-2.0	-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	

$k = 2$

Stupid policy



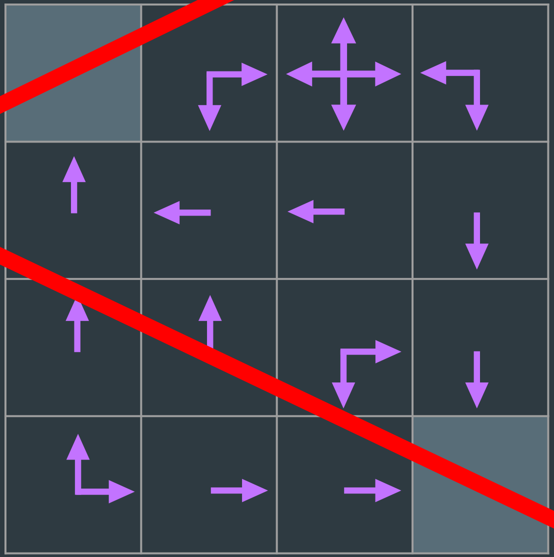
Policy Improvement in Grid-World

V_3 for random policy

$k = 3$

	-3.0	-3.0	-3.0
-2.4	-2.8	-3.0	-2.9
-2.9	-3.0	-2.9	-2.4
-3.0	-2.9	-2.4	

Stupid policy



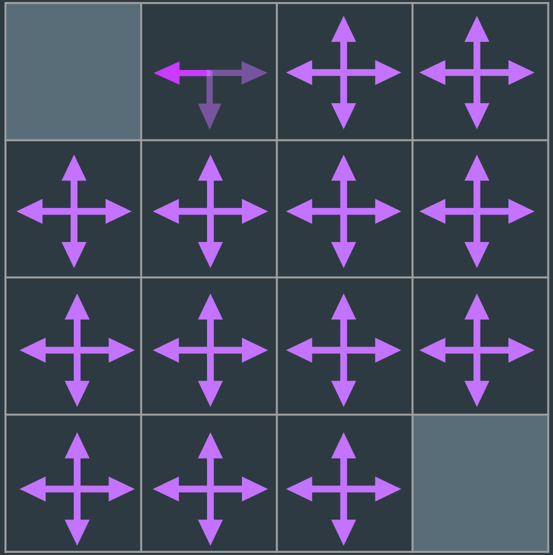
Policy Improvement in Grid-World

V_0 for 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	

Improved policy



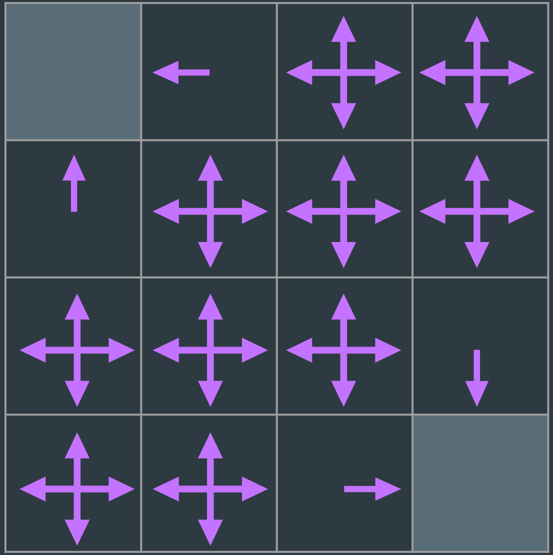
Policy Improvement in Grid-World

V_1 for random policy

	-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	

$k = 1$

Improved policy



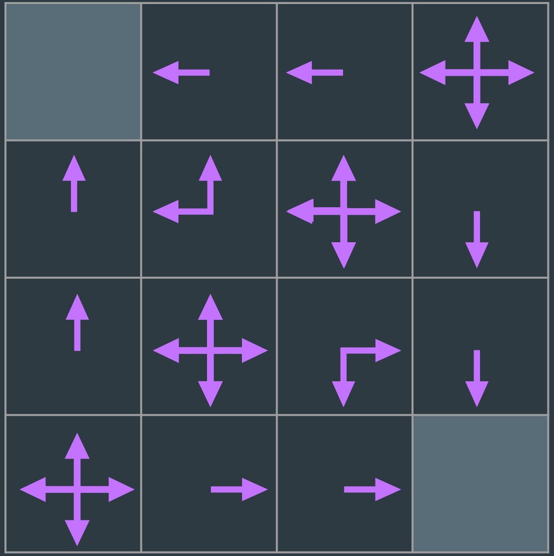
Policy Improvement in Grid-World

V_2 for random policy

	-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	

$k = 2$

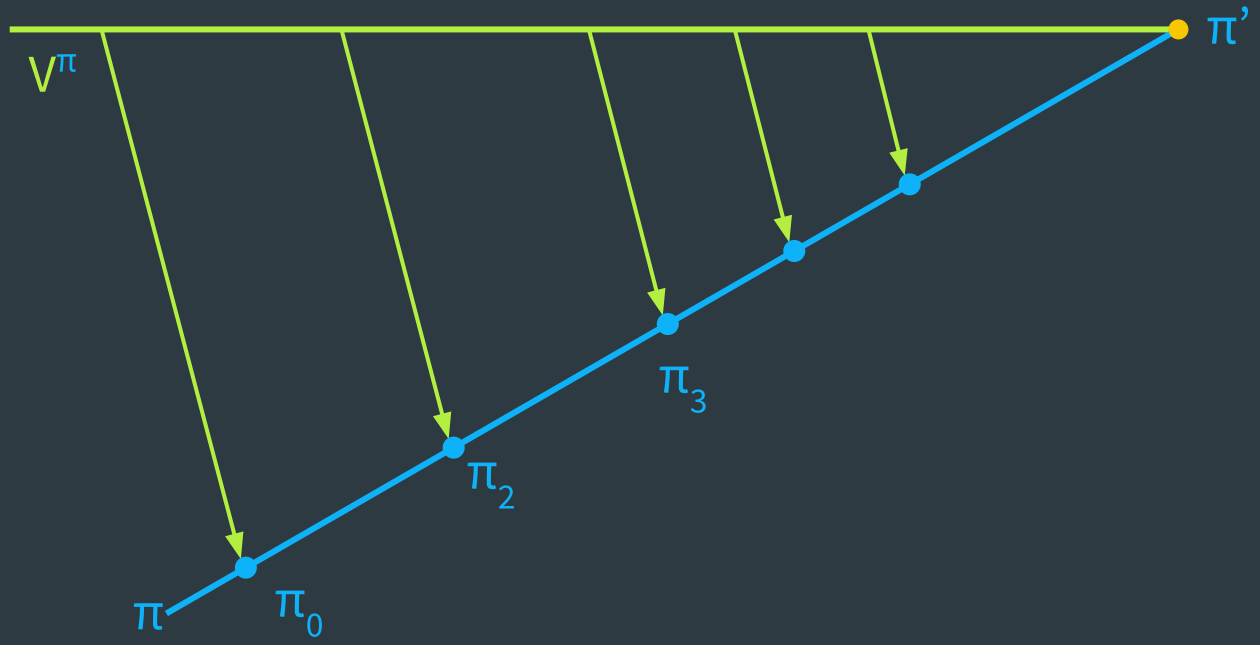
Improved policy



Policy Improvement Algorithm

```
policyImprove(π, isPolicyStable):

  isPolicyStable = true

  for s in S:
    oldAction = π[s]

    for a in A:
      Q[s][a] = sum(S', p(s', s, a) * (R(s', s, a) + V[s']))

    π[s] = argmax(A, Q[s])

    isPolicyStable = (oldAction is π[s])

  return {π, isPolicyStable} # approx. π*
```

Policy Improvement Visualization



Agenda

1. Last Week (Ch. 1, 3[1])
2. Bellman Optimality Principle (Ch. 3[1])
3. Planning (Ch. 4[1])
    a. Policy Evaluation
    b. Policy Improvement
    c. **Policy Iteration**

Side note: Value Iteration

For all s in S: $\{V_0, V_1, \ldots, V_k\}$ can be shown to converge to $V_*(s)$ by:

$$V_{t+1}(s) = \Sigma_a \pi(a \mid s) \times [R(s, a) + \Sigma_{s'} T(s, a, s') \times \gamma V_t(s')]$$

$$= \Sigma_a \pi(a \mid s) \times Q_t(s, a)$$

$$V_{t+1}(s) = \max_a Q^*(s, a)$$

Limitations of Value Iteration

- Can be tedious
- Convergence exactly to V^{π} only occurs in the limit
  - Grid-World hints of another method
- Want to minimize work needed to find π^*

Policy Iteration

How do we improve a policy π alongside iterating $V^{\pi}(s)$?

Policy Iteration

How do we improve a policy π alongside iterating $V^{\pi}(s)$?

Find better policy π' and use it to compute $V^{\pi'}(s)$

Policy Iteration

How do we improve a policy π alongside iterating $V^{\pi}(s)$?

Find better policy π' and use it to compute $V^{\pi'}(s)$

and use $V^{\pi'}(s)$ to compute π''

Policy Iteration

How do we improve a policy π alongside iterating $V^{\pi}(s)$?

Find better policy π' and use it to compute $V^{\pi'}(s)$

and use $V^{\pi'}(s)$ to compute π''

and use π'' to compute $V^{\pi''}(s)$

Policy Iteration

How do we improve a policy π alongside iterating $V^{\pi}(s)$?

Find better policy π' and use it to compute $V^{\pi'}(s)$

and use $V^{\pi'}(s)$ to compute π''

and use π'' to compute $V^{\pi''}(s)$

and use $V^{\pi''}(s)$ to compute π'''

Policy Iteration

How do we improve a policy π alongside iterating $V^{\pi}(s)$?

Find better policy π' and use it to compute $V^{\pi'}(s)$

and use $V^{\pi'}(s)$ to compute π''

and use π'' to compute $V^{\pi''}(s)$

and use $V^{\pi''}(s)$ to compute π'''

and use π''' to compute $V^{\pi'''}(s)$

Policy Iteration

How do we improve a policy π alongside iterating $V^{\pi}(s)$?

Find better policy π' and use it to compute $V^{\pi'}(s)$

$$\pi_0 \xrightarrow{E} V^{\pi_0} \xrightarrow{I} \pi_1 \xrightarrow{E} V^{\pi_1} \xrightarrow{I} \dots \xrightarrow{I} \pi^* \xrightarrow{E} V^*$$

Policy Iteration

How do we improve a policy π alongside iterating $V^{\pi}(s)$?

Find better policy π' and use it to compute $V^{\pi'}(s)$

Policy <u>improvement</u>

$$\pi_0 \xrightarrow{E} V^{\pi_0} \xrightarrow{I} \pi_1 \xrightarrow{E} V^{\pi_1} \xrightarrow{I} \cdots \xrightarrow{I} \pi^* \xrightarrow{E} V^*$$

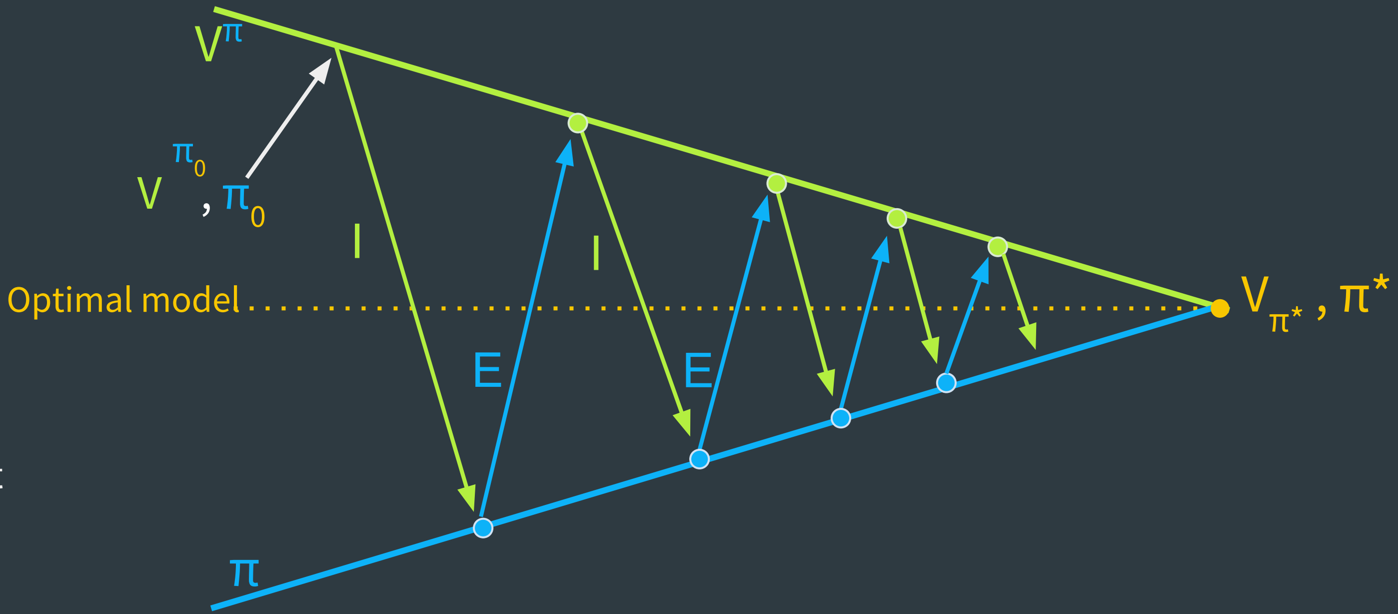
Policy <u>evaluation</u>

Policy Iteration Visualization

Think of $\pi_0 \xrightarrow{E} V^{\pi_0} \xrightarrow{I} \pi_1 \xrightarrow{E} V^{\pi_1} \xrightarrow{I} \dots \xrightarrow{I} \pi^* \xrightarrow{E} V^*$ as:

$\xrightarrow{E}$ Policy evaluation

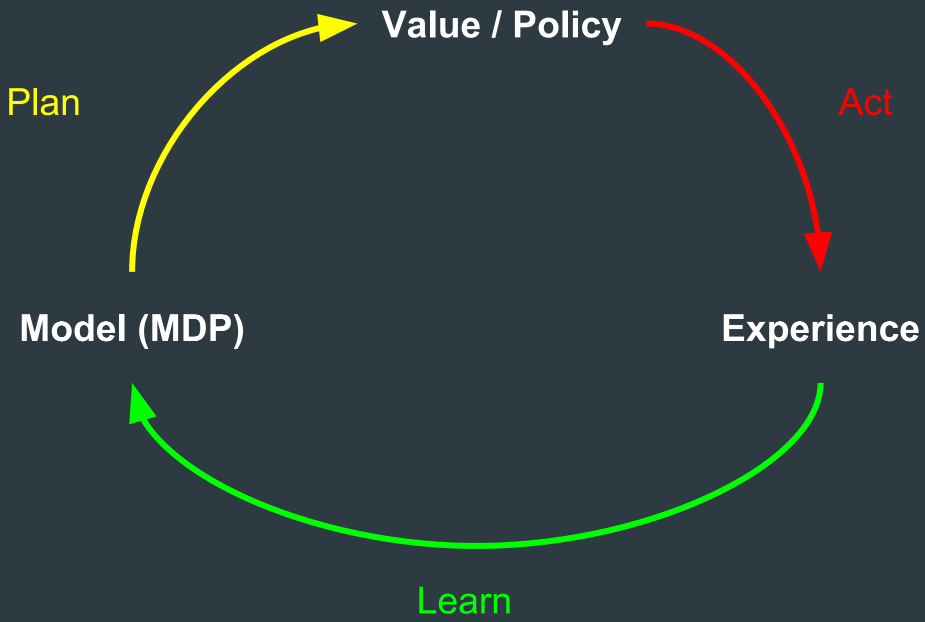
$\xrightarrow{I}$ Policy improvement



Policy Iteration Algorithm

```
policyIteration(S, A, π, V, Q, step):

  if step == 0:
    # V[s] in Real #s, π[s] in A
    init(V[s], π[s])

  try:
    V = policyEval(π)
  catch:
    # policy doesn't converge
    return policyIteration(S, A, π, V, Q, step)

  # π to π'
  π, policyStable = policyImprove(π, true)

  # return approx of V* and π*
  return {V, π} if policyStable else policyIteration(S, A, π, V, Q, step++)
```

Next Time: Learn



Questions?

References (Check 'em Out)

1. "Reinforcement Learning: An Introduction" - Sutton and Barto
2. CS188 at Berkeley

Resources (Check 'em Out)

- This Week in Machine Learning and AI (TWIMLAI)
- Mapping Babel
- Two Minute Papers
- Talking Machines
- CS188 at Berkeley by P. Abbeel
- Machine Learning on Coursera by A. Ng
- CS231 at Stanford by Fei-Fei Lei & Andrej Karpathy