Reinforcement Learning

Saurabh Gupta

Agent Environment Interface

Reinforcement Learning

Image Source: https://www.freecodecamp.org/news/a-brief-introduction-to-reinforcement-learning-7799af5840db/

Markov Decision Process

o_t

Step Back

a_t

o_{t+1}

…

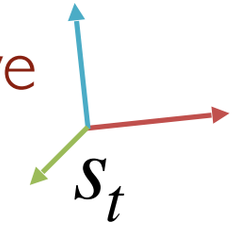
Transition Function

How you move, how the tiger moves?

Reward Function

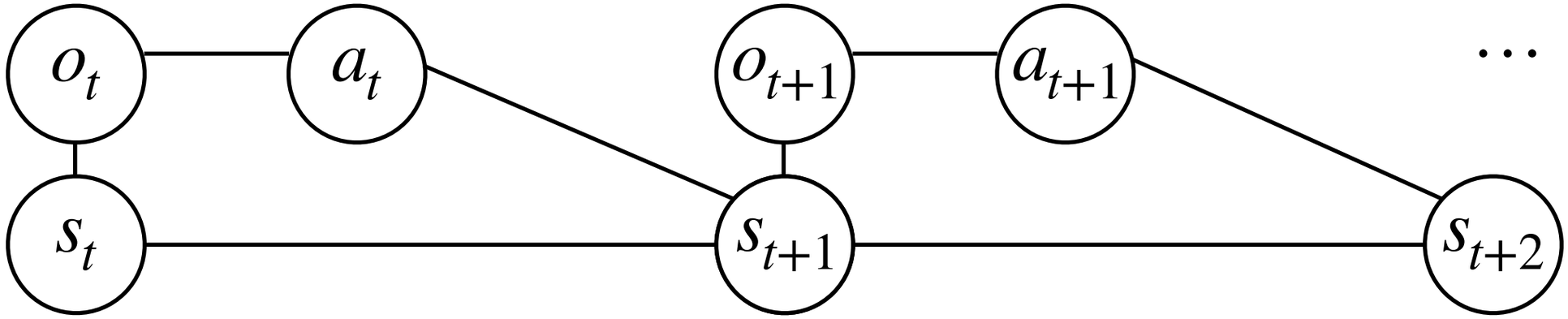
Survived?

3D Relative Pose



s_t

s_{t+1}



One step dynamics $p(s_{t+1}, r_{t+1} \mid s_t, a_t)$

Transition Function $p(s_{t+1} \mid s_t, a_t)$ $p(s_{t+2} \mid s_{t+1}, a_{t+1})$

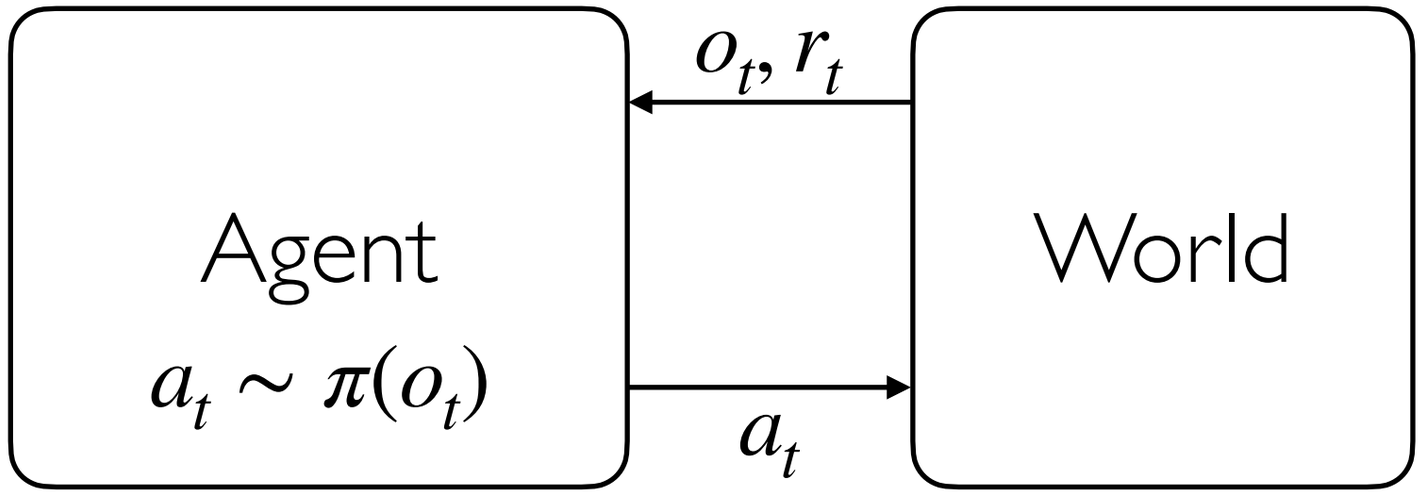
Reward Function $r_{t+1} = R(s_{t+1}, s_t, a_t)$ $r_{t+2} = R(s_{t+2}, s_{t+1}, a_{t+1})$

Goal $\text{argmax}_{a_0,\ldots,a_T} \sum_t \gamma^t r_t$

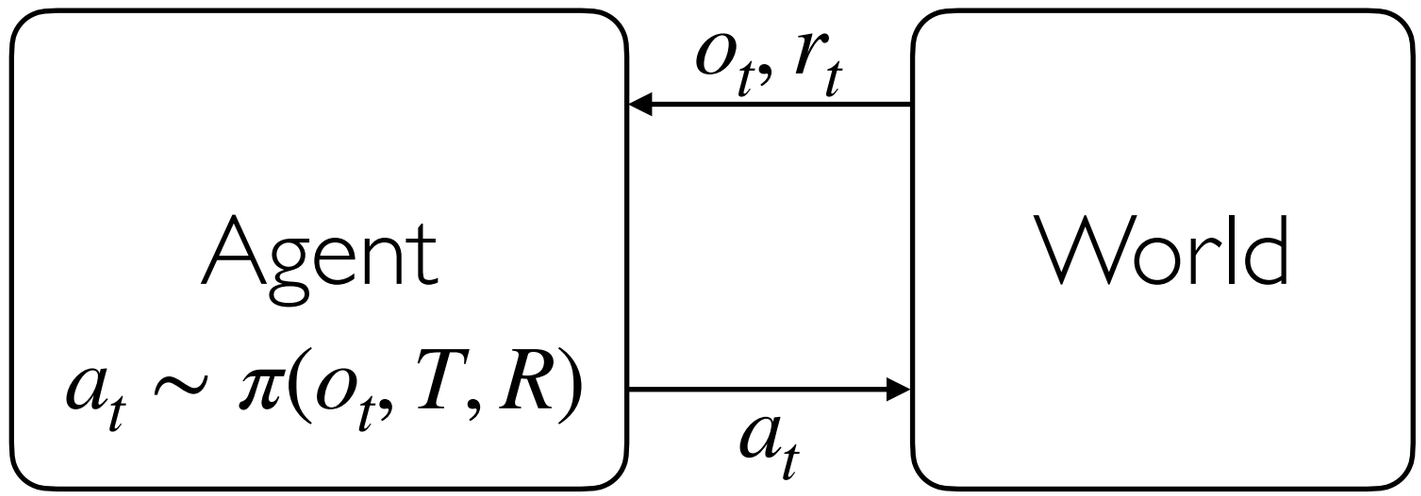
Solving MDPs

Policy: $a_t \sim \pi(o_t)$

Most General Case



More Specific Case



Fully Observed System $o_t = s_t$

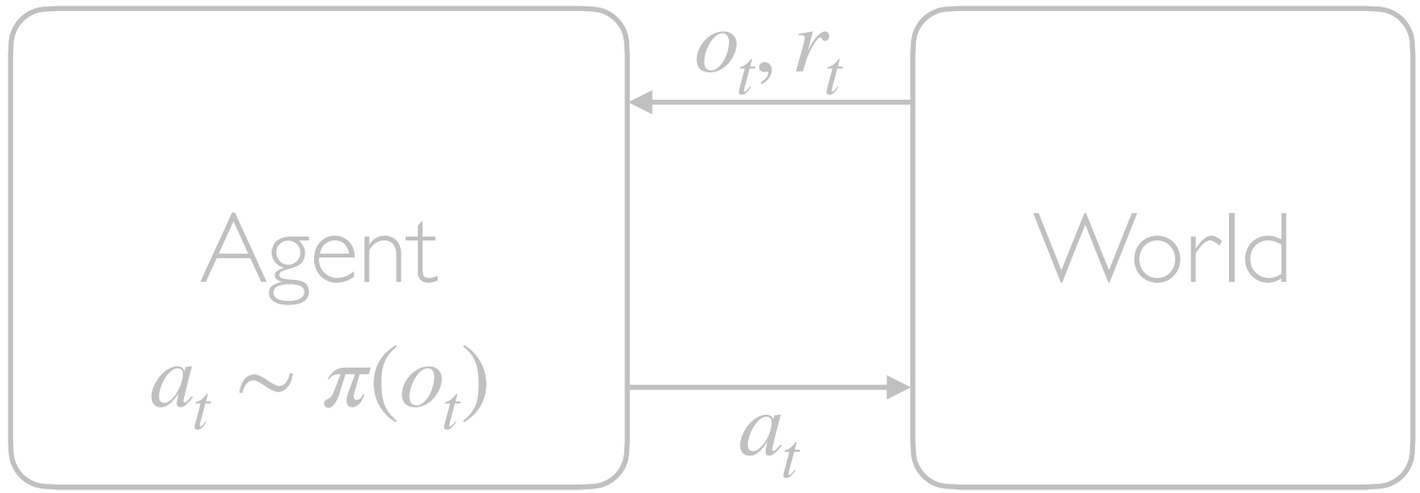
Known Transition Function $s_{t+1} \sim T(s_t, a_t)$

Known Reward Function $R(s_{t+1}, s_t, a_t)$

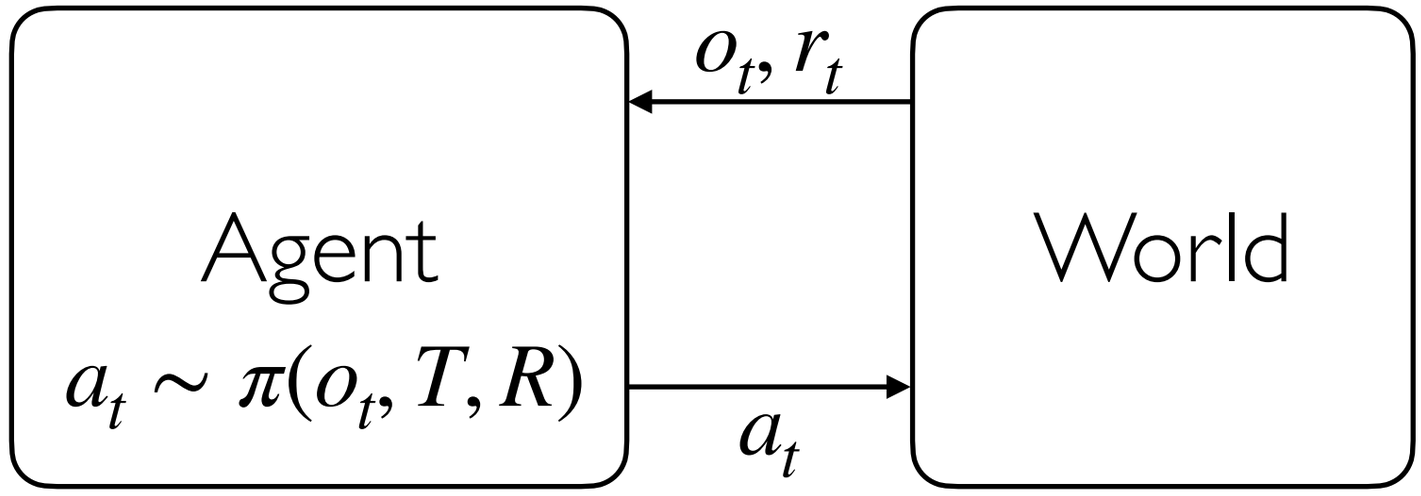
Solving MDPs

Policy: $a_t \sim \pi(o_t)$

Most General Case



More Specific Case



Fully Observed System $o_t = s_t$

Known Transition Function $s_{t+1} \sim T(s_t, a_t)$

Known Reward Function $R(s_{t+1}, s_t, a_t)$

Basics

Policy

Episodes

Returns

Value Functions

Action-value Functions

Solving MDPs via Dynamic Programming

Policy Evaluation

Policy Improvement

Policy Iteration

Value Iteration

Iterative Policy Evaluation

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 theorem

- Given the value function for *any policy* π

$$q_\pi(s, a) \qquad \text{for all } s, a$$

- It can always be greedified to obtain a *better policy*:

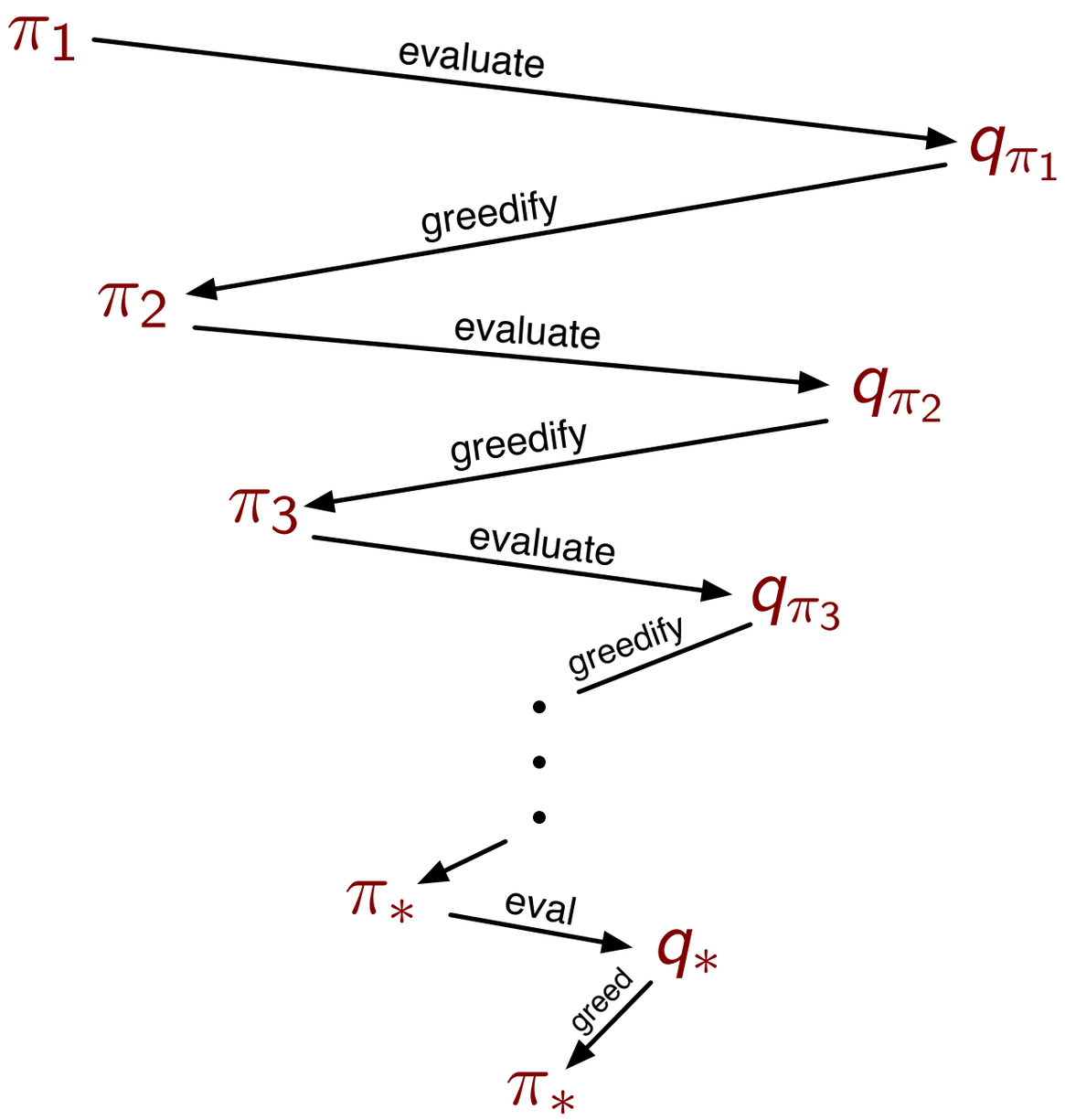
$$\pi'(s) = \arg\max_a q_\pi(s, a) \qquad (\pi' \text{ is not unique})$$

- where better means:

$$q_{\pi'}(s, a) \geq q_\pi(s, a) \qquad \text{for all } s, a$$

- with equality only if both policies are optimal

The dance of policy and value (Policy Iteration)



Any policy evaluates to a unique value function (soon we will see how to learn it)

which can be greedified to produce a better policy

That in turn evaluates to a value function

which can in turn be greedified…

Each policy is *strictly better* than the previous, until *eventually both are optimal*

There are *no local optima*

The dance converges in a finite number of steps, usually very few

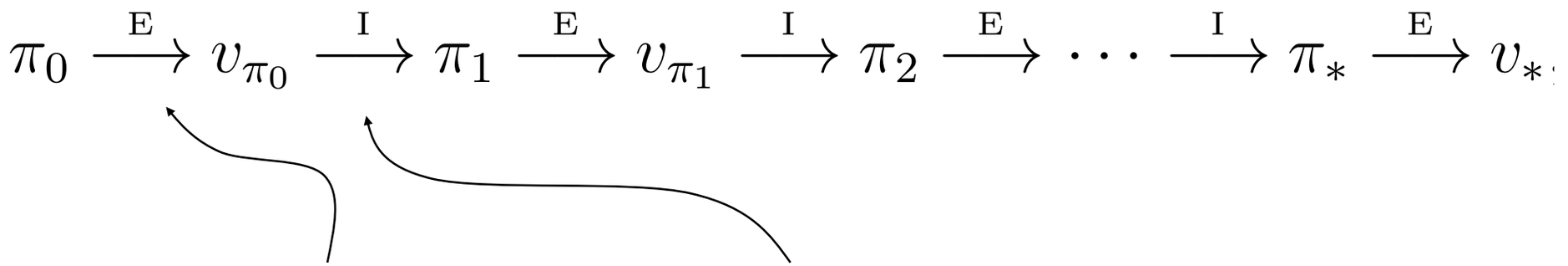
Policy Iteration

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 $s \in \mathcal{S}$:
      $a \leftarrow \pi(s)$
      $\pi(s) \leftarrow \arg\max_a \sum_{s',r} p(s', r|s, a)\big[r + \gamma V(s')\big]$
      If $a \neq \pi(s)$, then *policy-stable* $\leftarrow$ *false*
   If *policy-stable*, then stop and return V and π; else go to 2

Policy Iteration



Value Iteration

Recall the **full policy-evaluation backup**:

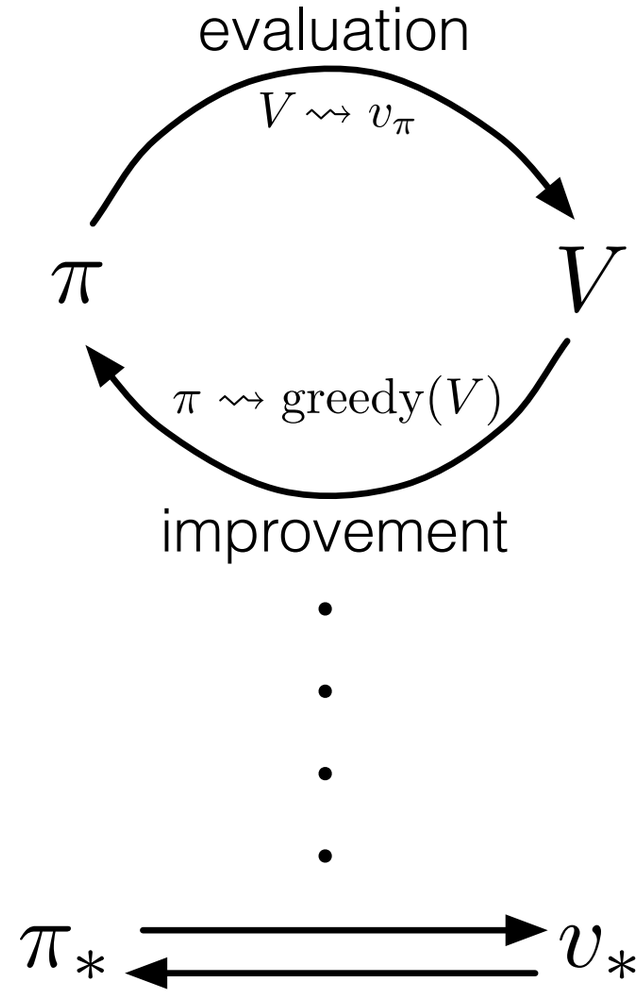
$$v_{k+1}(s) = \sum_a \pi(a|s) \sum_{s',r} p(s',r|s,a)\Big[r + \gamma v_k(s')\Big] \qquad \forall s \in \mathcal{S}$$

Here is the **full value-iteration backup**:

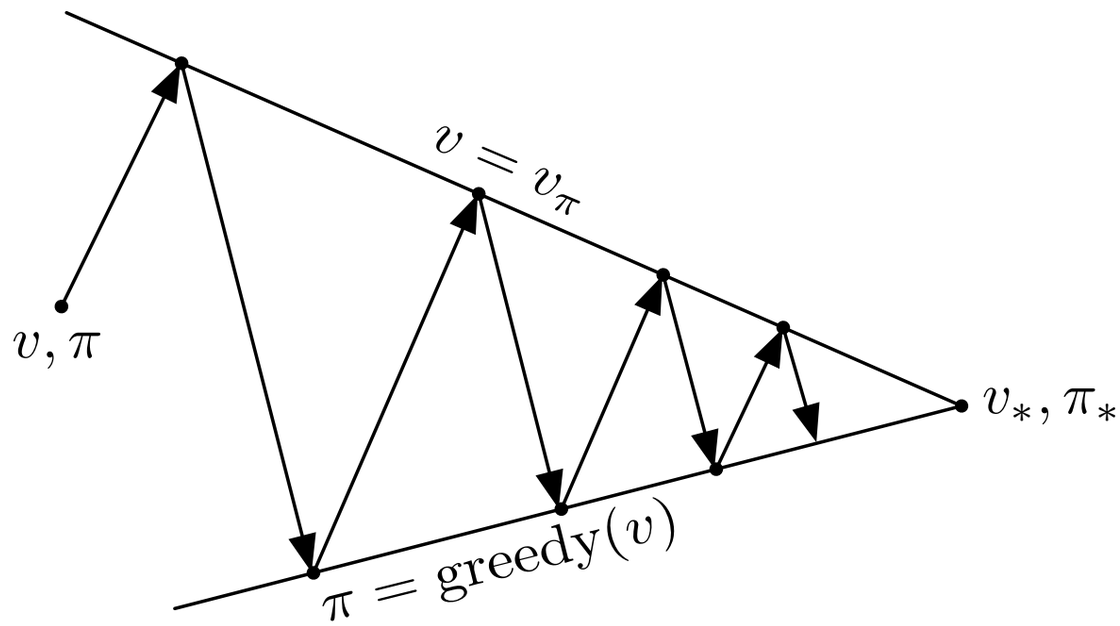
$$v_{k+1}(s) = \max_a \sum_{s',r} p(s',r|s,a)\Big[r + \gamma v_k(s')\Big] \qquad \forall s \in \mathcal{S}$$

Generalized Policy Iteration

Generalized Policy Iteration (GPI):
any interaction of policy evaluation and policy improvement, independent of their granularity.



A geometric metaphor for convergence of GPI:



Value Iteration

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

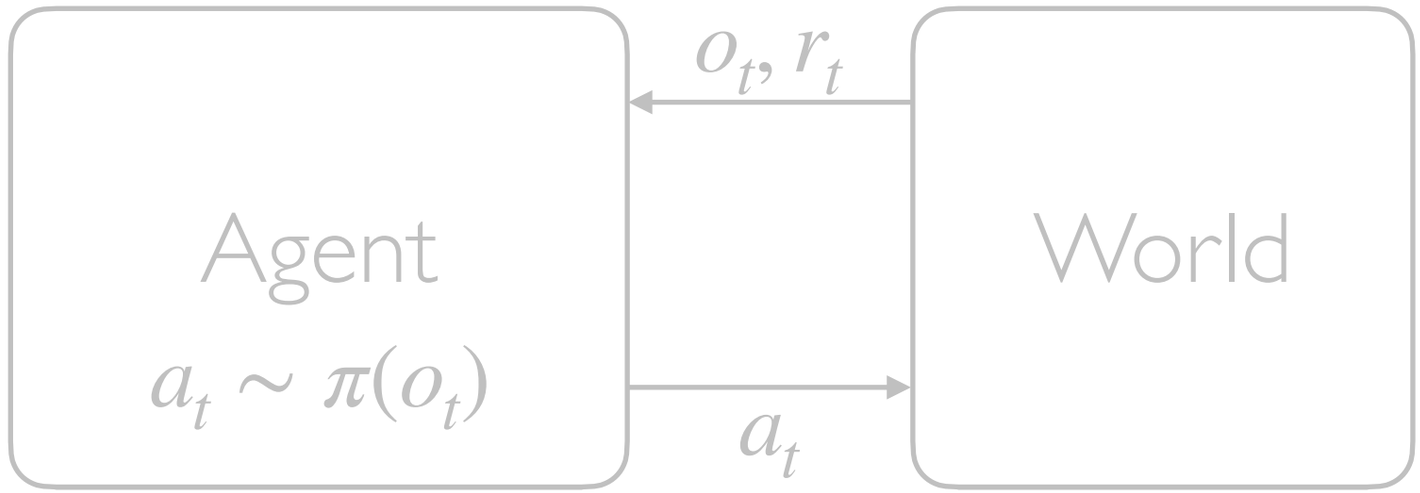
Repeat
  $\Delta \leftarrow 0$
  For each $s \in \mathcal{S}$:
    $v \leftarrow V(s)$
    $V(s) \leftarrow \max_a \sum_{s',r} p(s', r|s, a)\big[r + \gamma V(s')\big]$
    $\Delta \leftarrow \max(\Delta, |v - V(s)|)$
until $\Delta < \theta$ (a small positive number)

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

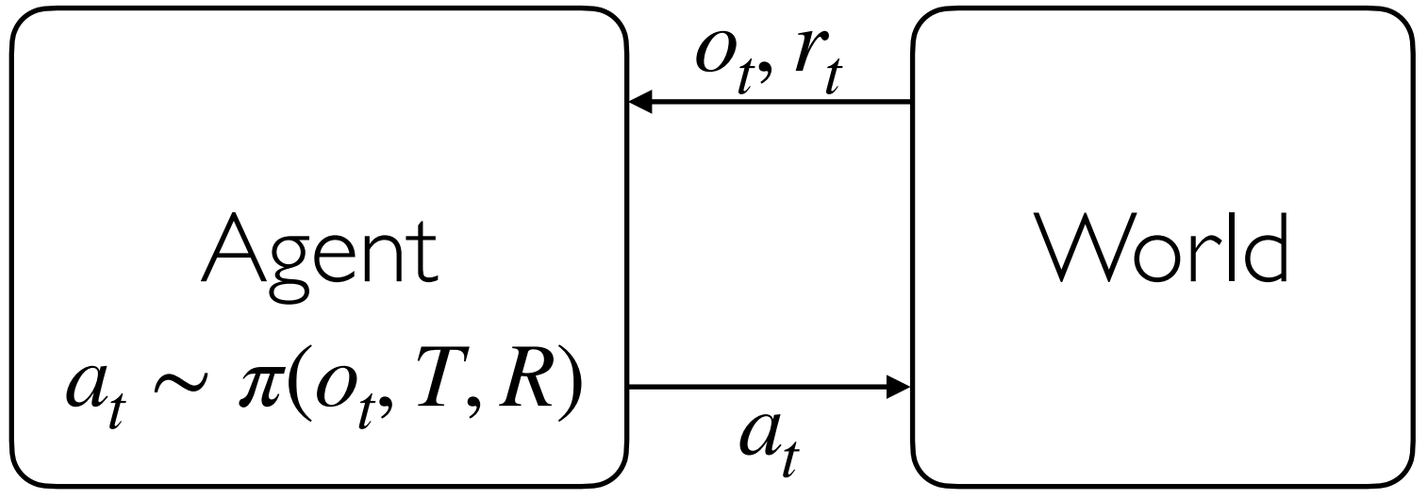
Solving MDPs

Policy: $a_t \sim \pi(o_t)$

Most General Case



More Specific Case



Fully Observed System $o_t = s_t$

Known Transition Function $s_{t+1} \sim T(s_t, a_t)$

Known Reward Function $R(s_{t+1}, s_t, a_t)$

Resources

Reinforcement Learning: An Introduction
Sutton and Barto
http://incompleteideas.net/book/the-book-2nd.html

David Silver's **Reinforcement Learning Course**
https://www.davidsilver.uk/teaching/

Thank you