Reinforcement Learning: Bellman Equations and Dynamic Programming

B. Ravindran

Value Functions(Recall)

- The **value of a state** is the expected long-term reward starting from that state; depends on the agent's policy:

$$v_\pi(s) \doteq \mathbb{E}_\pi[G_t \mid S_t = s] = \mathbb{E}_\pi\left[\sum_{k=0}^{\infty} \gamma^k R_{t+k+1} \,\middle|\, S_t = s\right], \text{ for all } s \in \mathcal{S},$$

Bellman Equation for a Policy π

$$
\begin{aligned}
v_\pi(s) &\doteq \mathbb{E}_\pi[G_t \mid S_t = s] \\
&= \mathbb{E}_\pi[R_{t+1} + \gamma G_{t+1} \mid S_t = s] \\
&= \sum_a \pi(a|s) \sum_{s'} \sum_r p(s', r|s, a) \Big[r + \gamma \mathbb{E}_\pi[G_{t+1} | S_{t+1} = s'] \Big] \\
&= \sum_a \pi(a|s) \sum_{s', r} p(s', r|s, a) \Big[r + \gamma v_\pi(s') \Big], \quad \text{for all } s \in \mathcal{S},
\end{aligned}
$$

- Linear equation in $|S|$ variables
- Unique solution exists

Optimal Value Functions

- For finite MDPs, policies can be partially ordered:

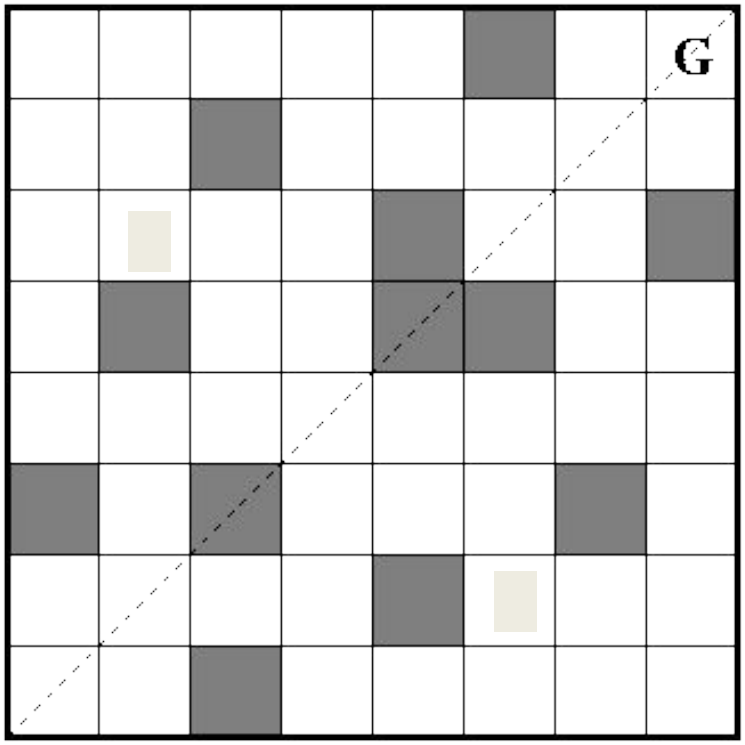
$$\pi \geq \pi' \text{ if and only if } v_\pi(s) \geq v_{\pi'}(s') \text{ for all } s \in \mathcal{S}$$

- There is always at least one (and possibly many) policies that is better than or equal to all the others. This is an optimal policy. We denote them all π_*
- Optimal policies share the same optimal state-value function:

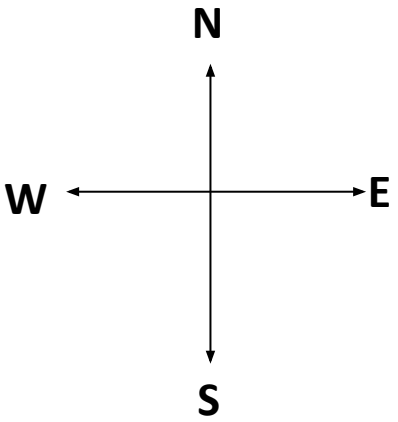
$$v_*(s) = \max_\pi v_\pi(s) \text{ for all } s \in \mathcal{S}$$

$$q_*(s,a) = \max_\pi q_\pi(s,a) \text{ for all } s \in \mathcal{S} \text{ and } a \in \mathcal{A}$$

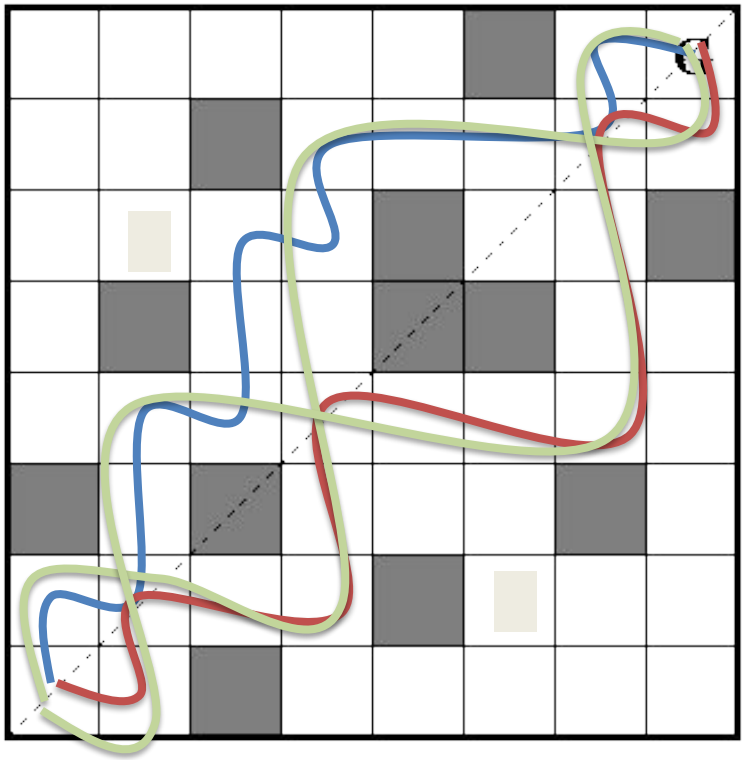
Example



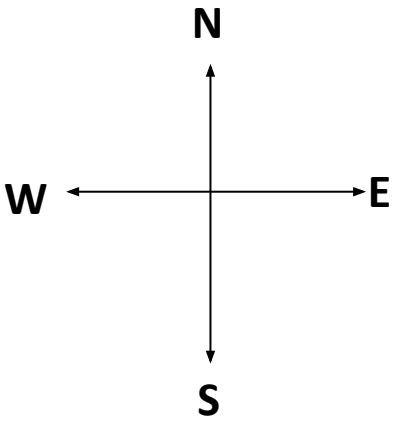
$$M = \langle S, A, p, r \rangle$$



Example



$$M = \langle S, A, p, r \rangle$$



Many optimal policies but only one optimal value function

Bellman Optimality Equation for v_*

The value of a state under an optimal policy must equal the expected return for the best action from that state:

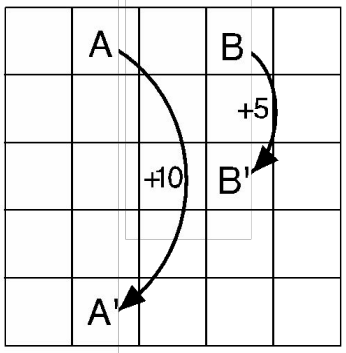
$$
\begin{aligned}
v_*(s) &= \max_{a \in \mathcal{A}(s)} q_{\pi_*}(s,a) \\
&= \max_a \mathbb{E}_{\pi_*}[G_t \mid S_t = s, A_t = a] \\
&= \max_a \mathbb{E}_{\pi_*}[R_{t+1} + \gamma G_{t+1} \mid S_t = s, A_t = a] \\
&= \max_a \mathbb{E}[R_{t+1} + \gamma v_*(S_{t+1}) \mid S_t = s, A_t = a] \\
&= \max_a \sum_{s',r} p(s', r \mid s, a)\left[r + \gamma v_*(s')\right].
\end{aligned}
$$

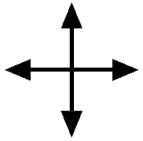
Bellman Optimality Equation for q_*

$$
\begin{aligned}
q_*(s,a) &= \mathbb{E}\left[R_{t+1} + \gamma \max_{a'} q_*(S_{t+1}, a') \;\middle|\; S_t = s, A_t = a\right] \\
&= \sum_{s',r} p(s', r \,|\, s, a)\left[r + \gamma \max_{a'} q_*(s', a')\right].
\end{aligned}
$$

Another Example

- Actions: `north`, `south`, `east`, `west`; deterministic.
- If would take agent off the grid: no move but reward = –1
- Other actions produce reward = 0, except actions that move agent out of special states A and B as shown.





3.3	8.8	4.4	5.3	1.5
1.5	3.0	2.3	1.9	0.5
0.1	0.7	0.7	0.4	-0.4
-1.0	-0.4	-0.4	-0.6	-1.2
-1.9	-1.3	-1.2	-1.4	-2.0

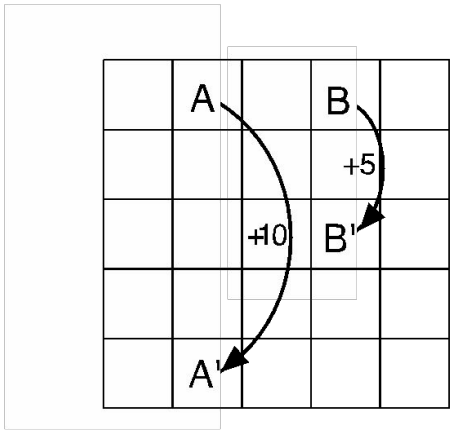
State-value function for equiprobable random policy; $\gamma = 0.9$

Why Optimal State-Value Functions are Useful

Any policy that is greedy with respect to v_* is an optimal policy.

Therefore, given v_*, one-step-lookahead search produces the long-term optimal actions.

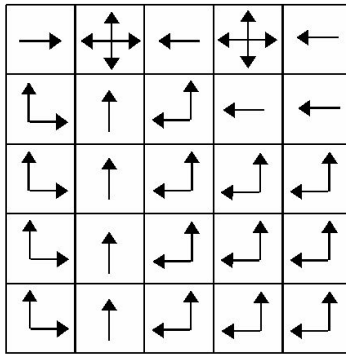
E.g., back to the gridworld:



a) gridworld

22.0	24.4	22.0	19.4	17.5
19.8	22.0	19.8	17.8	16.0
17.8	19.8	17.8	16.0	14.4
16.0	17.8	16.0	14.4	13.0
14.4	16.0	14.4	13.0	11.7

b) V^*



c) π^*

What About Optimal Action-Value Functions?

Given q_* , the agent does not even have to do a one-step-ahead search:

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

Dynamic Programming

- DP is the solution method of choice for MDPs
  - Requires complete knowledge of system dynamics (transition matrix and rewards)
  - Computationally expensive
  - Curse of dimensionality
  - Guaranteed to converge!

Policy Evaluation

- For a given policy π, compute the state value function v_π
- Recall Bellman equation for v_π:

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

  - a system of $|S|$ simultaneous linear equations
  - solve iteratively

Iterative Policy Evaluation Algo.

Input π, the policy to be evaluated
Algorithm parameter: a small threshold $\theta > 0$ determining accuracy of estimation
Initialize $V(s)$, for all $s \in \mathcal{S}^+$, arbitrarily except that $V(terminal) = 0$

Loop:
  $\Delta \leftarrow 0$
  Loop 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 $\Delta < \theta$

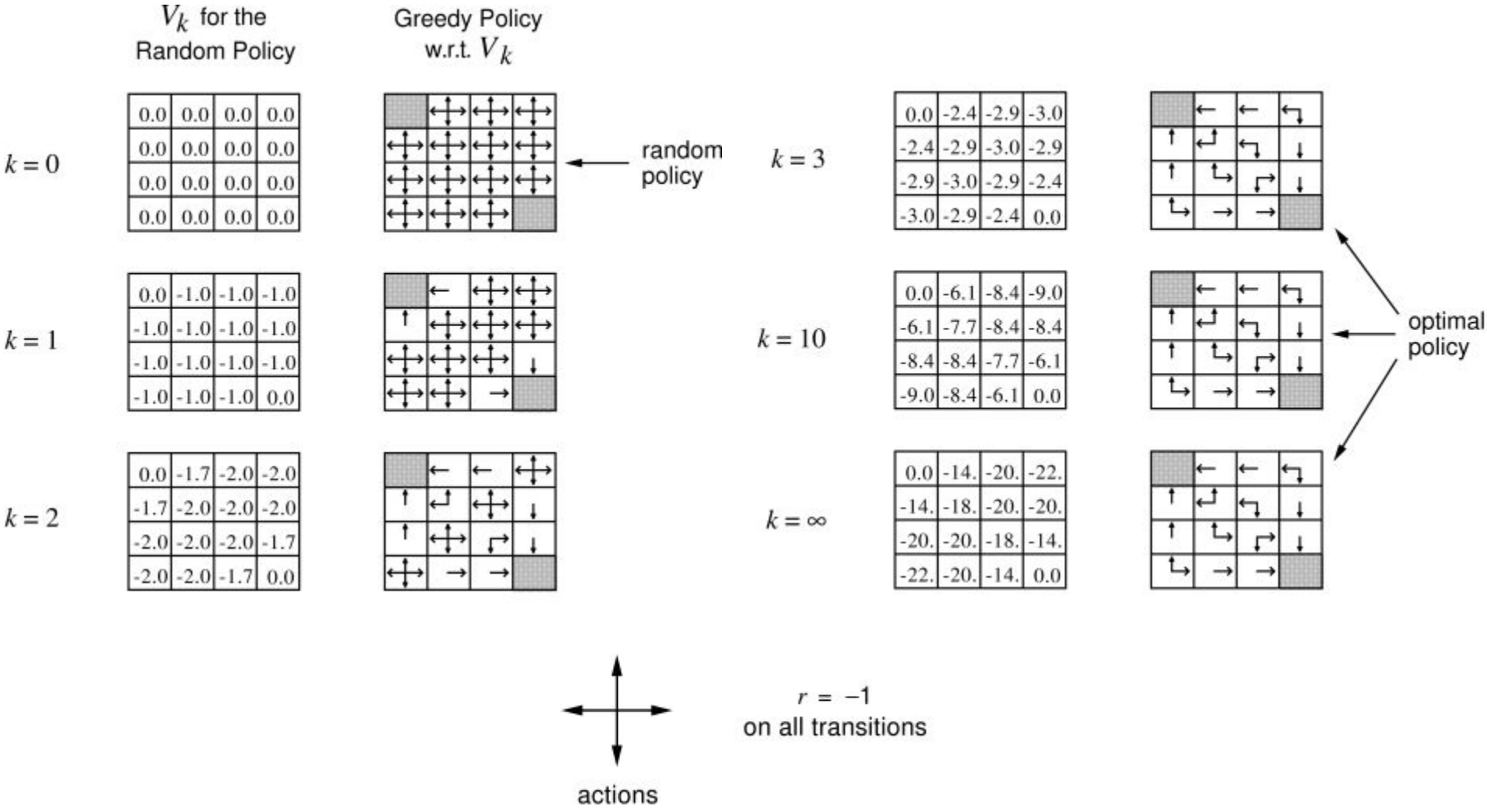
The Bellman Operator T_π

- In the previous algo, the update to $V(s)$ can be interpreted as an operator acting on a vector V

$$T_\pi : \mathbb{R}^{|S|} \rightarrow \mathbb{R}^{|S|}$$

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

Example of Policy Evaluation



Policy Improvement

- Suppose we have computed v_π for an arbitrary deterministic policy π
- For a given state s, would it be better to choose an action $a \neq \pi(s)$?
- The value of doing a in state s is:

$$\begin{aligned} q_\pi(s,a) &\doteq \mathbb{E}[R_{t+1} + \gamma v_\pi(S_{t+1}) \mid S_t = s, A_t = a] \\ &= \sum_{s',r} p(s', r \mid s, a)\Big[r + \gamma v_\pi(s')\Big]. \end{aligned}$$

- It is better to switch to action a for state s if and only if

$$q_\pi(s,a) > v_\pi(s)$$

Policy Improvement Cont.

Do this for all states to get a new policy π' that is **greedy** with respect to v_π:

$$\begin{aligned}\pi'(s) &= \arg\max_a q_\pi(s,a) \\ &= \arg\max_a \sum_{s'} p(s', r \mid s, a)\,[r + \gamma v_\pi(s')]\end{aligned}$$

Then, $v_{\pi'} \geq v_\pi$

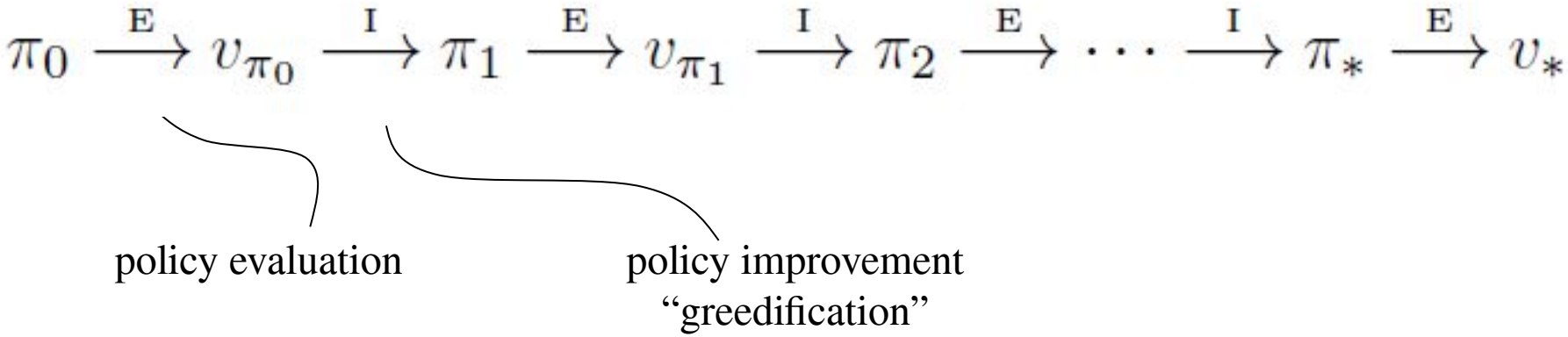
Policy Improvement Cont.

What if $v_{\pi'} = v_\pi$? Then, for all $s, \in \mathcal{S}$, we have

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

But this is the Bellman Optimality equation.
So $v_{\pi'} = v_*$ and both π and π' are optimal policies.

Policy Iteration



Policy Iteration Algo.

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

2. Policy Evaluation
   Loop:
   $\Delta \leftarrow 0$
   Loop for each $s \in \mathcal{S}$:
   $v \leftarrow V(s)$
   $V(s) \leftarrow \sum_{s',r} p(s', r \mid s, \pi(s)) \left[r + \gamma V(s') \right]$
   $\Delta \leftarrow \max(\Delta, |v - V(s)|)$
   until $\Delta < \theta$ (a small positive number determining the accuracy of estimation)

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

Value Iteration

- Policy evaluation step of policy iteration can be truncated without losing convergence.
- If policy evaluation step is stopped after one update of each state, we get value iteration
- Can also be interpreted as turning the Bellman optimality equation into an update rule.

$$
\begin{aligned}
v_{k+1}(s) &\doteq \max_a \mathbb{E}[R_{t+1} + \gamma v_k(S_{t+1}) \mid S_t = s, A_t = a] \\
&= \max_a \sum_{s',r} p(s', r \mid s, a) \Big[r + \gamma v_k(s') \Big],
\end{aligned}
$$

Value iteration Algo.

Algorithm parameter: a small threshold $\theta > 0$ determining accuracy of estimation
Initialize $V(s)$, for all $s \in \mathcal{S}^+$, arbitrarily except that $V(terminal) = 0$

Loop:
| $\Delta \leftarrow 0$
| Loop for each $s \in \mathcal{S}$:
| $v \leftarrow V(s)$
| $V(s) \leftarrow \max_a \sum_{s',r} p(s', r|s, a)\left[r + \gamma V(s')\right]$
| $\Delta \leftarrow \max(\Delta, |v - V(s)|)$
until $\Delta < \theta$

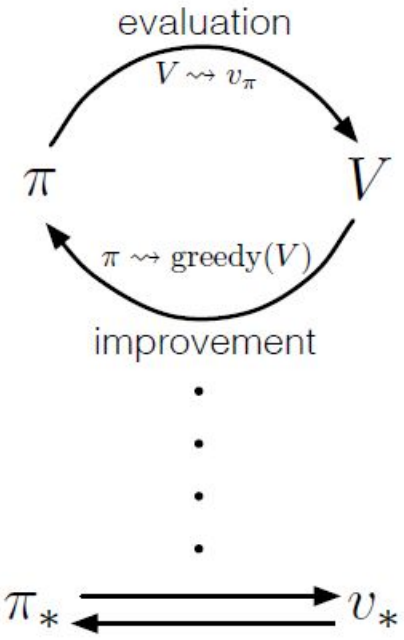
Output a deterministic policy, $\pi \approx \pi_*$, such that
$\pi(s) = \arg\max_a \sum_{s',r} p(s', r|s, a)\left[r + \gamma V(s')\right]$

Asynchronous DP

- Disadvantage of algorithms discussed is we have to do the updates over the entire state set.
- In asynchronous DP, the updates are not done over the entire state set at each iteration.
- Have to ensure that every state is visited sufficiently often for convergence.
- Gives flexibility to choose order of updates.
- Can intertwine real time interaction with the environment and DP updates.
- Can focus updates on parts of state space relevant to agent.

Generalized Policy Iteration

- GPI refers to the idea of letting policy evaluation and policy improvement interact, independent of their granularity.



GPI

- Almost all RL methods can be viewed as GPI.
- For example, policy iteration has evaluation running to completion before improvement begins. In value iteration, only one step of evaluation is done before the improvement step. In Asynchronous DP, the two are interleaved at a finer granularity.