Lecture 4:
Bellman Equations and Dynamic Programming

B. Ravindran

Value Functions (Recall)

The **value of a state** is the is the expected return when starting in state s and following π thereafter

$$v_\pi(s) \doteq \mathbb{E}_\pi[G_t | S_t = s]$$

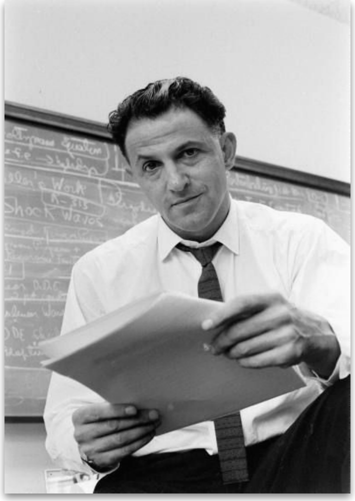
$$v_\pi(s) \doteq \mathbb{E}_\pi[\sum_{k=0}^{\infty} \gamma^k R_{t+k+1} | S_t = s]$$

Bellman Equation for a Policy π

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

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

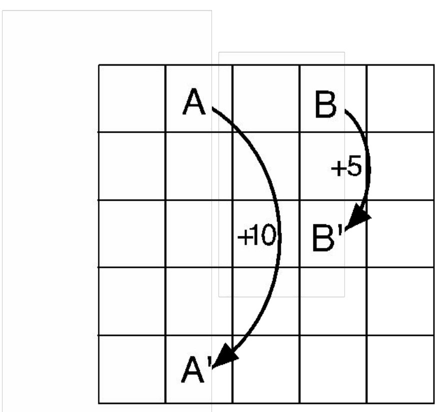
- ❑ Linear equation in |S| variables
- ❑ An unique solution exists

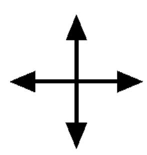


$$
q_\pi(s, a) \doteq \sum_{s',r} p(s', r|s, a)[r + \gamma \sum_{a'} \pi(a'|s') q(s', a')]
$$

An Example

- Actions: north, south, east, west *(deterministic)*
- If action 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	2.3	1.9	0.5
0.1	0.7	0.7	0.4	-0.4
-1	-0.4	-0.4	-0.6	-1.2
-1.9	-1.3	-1.2	-1.4	-2

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

Optimal Value Functions

- For finite MDPs, policies can be partially ordered:

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

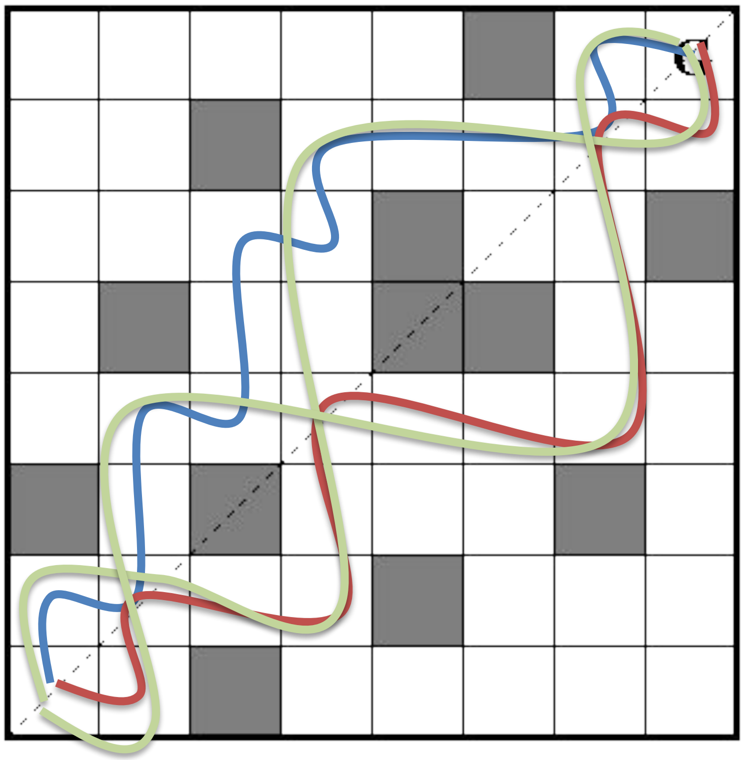
- There is always at least one *(and possibly many)* policies that is better than or equal to all the others. This is an optimal policy. We denote them all π_*

- Optimal policies share the same optimal state-value function:

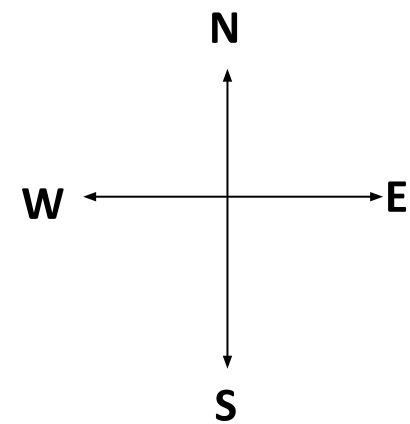
$$v_*(s) \doteq \max_\pi v_\pi(s) \quad \text{for all } s \in \mathcal{S}$$

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

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:

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

Bellman Optimality Equation for q_*

The expected return for taking action *a* in state *s* and thereafter following an optimal policy

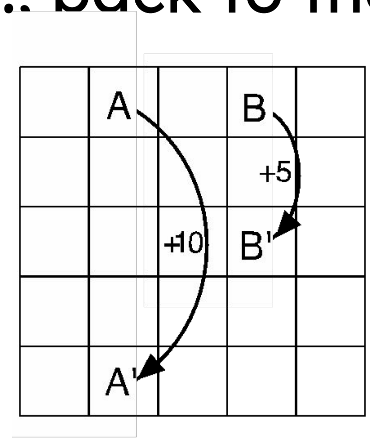
$$
\begin{aligned}
q_*(s,a) &= \mathbb{E}\Big[R_{t+1} + \gamma \max_{a'} q_*(S_{t+1},a') \;\Big|\; S_t = s, A_t = a\Big] \\
&= \sum_{s',r} p(s',r \,|\, s,a)\Big[r + \gamma \max_{a'} q_*(s',a')\Big].
\end{aligned}
$$

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.

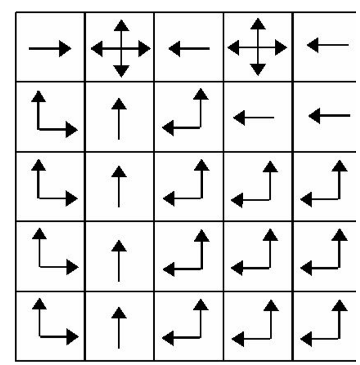
  E.g.. back to the gridworld:



a) gridworld

0.0	-14.0	-20.0	-22.0
-14.0	-18.0	-20.0	-20.0
-20.0	-20.0	-18.0	-14.0
-22.0	-20.0	-14.0	0.0

b) V^*



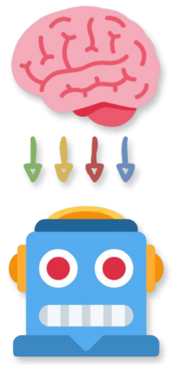
c) π^*

Why Optimal Action-Value Functions are More Useful?

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

$$\pi_*(s) = argmax_{a \in A(s)}\, q_*(s,a)$$

$$\pi_*(s) = argmax_{a \in A(s)} \cancel{\sum_{s',r}} p(s',r|s,a)[r + \gamma v_*(s')]$$



Dynamic Programming

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:

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

- A system of |S| simultaneous linear equations
- Solve iteratively

Iterative Policy Evaluation

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

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

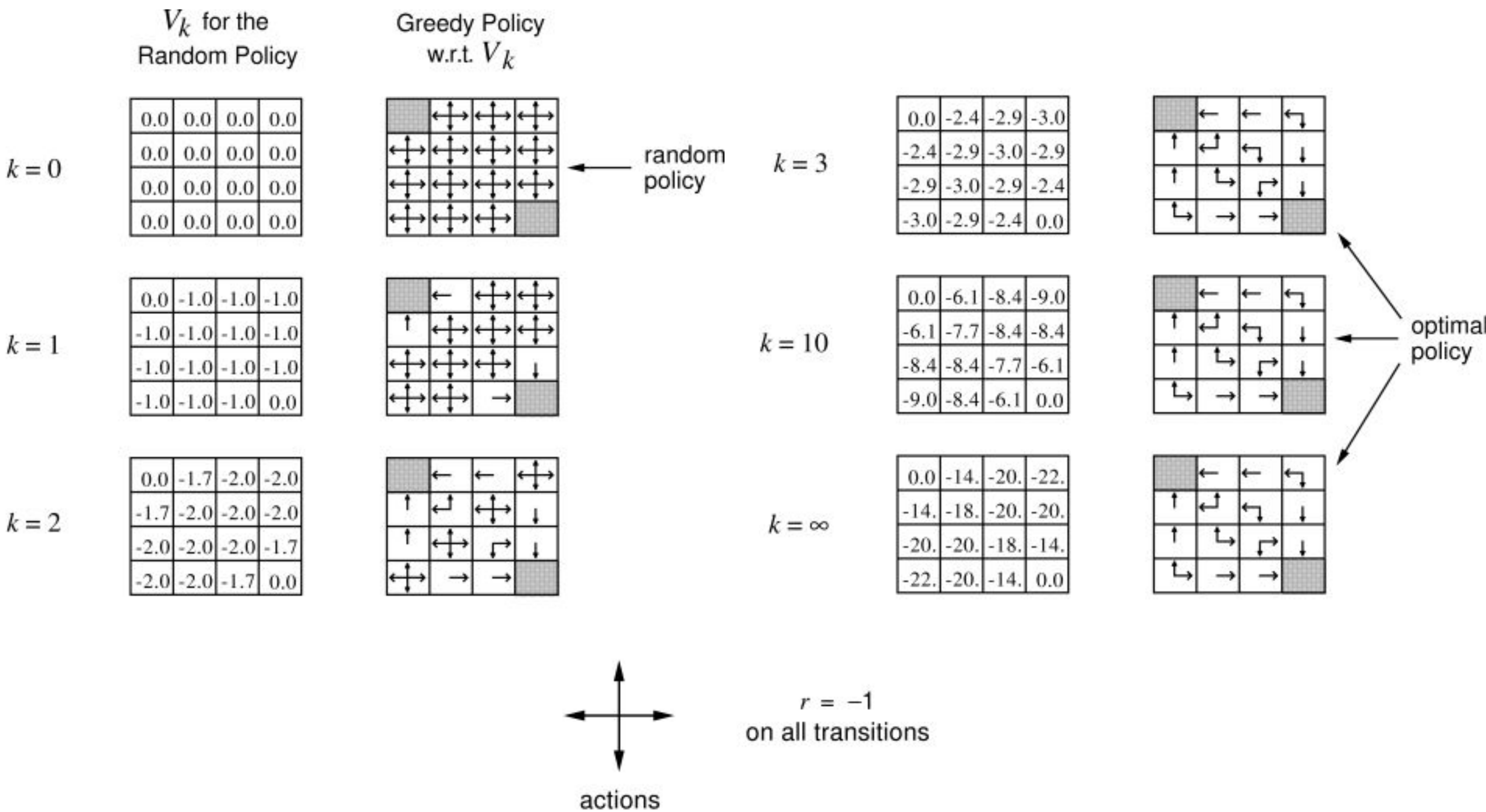
The Bellman Operator T_π

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

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

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

Example of Policy Evaluation



Policy Improvement

- Suppose we have computed v_π for an arbitrary *deterministic policy π*
- Question: 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:

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

- 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

$$\pi'(s) = \arg\max_a q_\pi(s,a)$$
$$= \arg\max_a \sum_{s',r} p(s', r|s,a)[r + \gamma v_\pi(s')]$$
$$Then, v_{\pi'} \geq v_\pi$$

Policy Improvement Cont.

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

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

But this is the Bellman Optimality equation.

So $v_{\pi'} = v_*$ and both π and π' are optimal policies.

Policy Iteration

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

Policy Evaluation

Policy Improvement
(greedification)

Policy Iteration Algo.

1. Initialization
   $V(s) \in \mathbb{R}$ and $\pi(s) \in \mathcal{A}(s)$ arbitrarily for all $s \in \mathcal{S}$; $V(terminal) \doteq 0$

2. Policy Evaluation
   Loop:
   $\quad \Delta \leftarrow 0$
   $\quad$ Loop 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 determining the accuracy of estimation)

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

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.

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

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

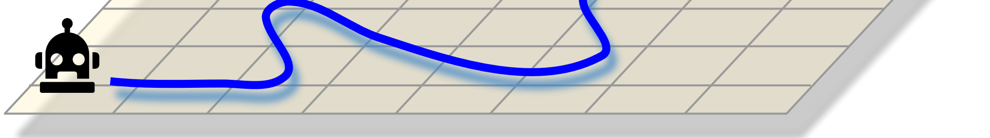
Output a deterministic policy, $\pi \approx \pi_*$, such that
$\pi(s) = \arg\max_a \sum_{s',r} p(s', r \mid s, a)\big[r + \gamma V(s')\big]$

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

Real-Time DP (RTDP)

- ❑ On-policy trajectory-sampling version of value-iteration algorithm.
- ❑ Updates values of states visited in the actual trajectory

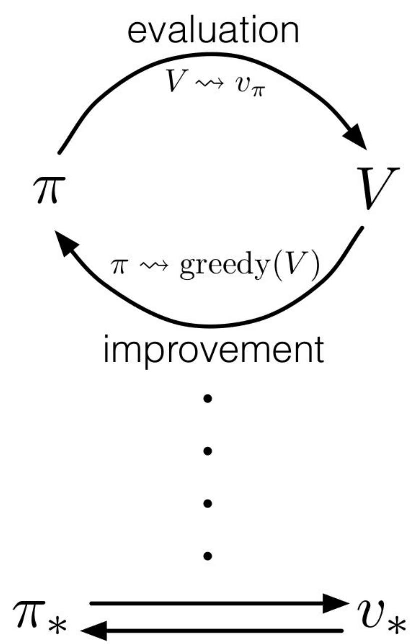


1. Take action according to π
2. Update $V_\pi(s)$
3. Update $\pi(a|s)$

- ❑ Unlike asynchronous-DP, no requirement to update every state infinitely often.

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.

- 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.