How to solve MDPs ?

One way is Dynamic Programming

↳ policy evaluation

↳ policy iteration

↳ value iteration

DP → break problem into subproblem

solve them & concatenate

Bellman equation gives

recursive decomposition

Dynamic Programming assumes full knowledge

of MDP, used for planning

for prediction MDP $\langle S, A, P, R, \gamma \rangle + \pi$ → $\boxed{IPE}$ → value function v_π

for control MDP $\langle S, A, P, R, \gamma \rangle$ → $\boxed{}$ → optimal value function v_* & optimal policy π_*

Policy Evaluation

method: iterative application of Bellman expectation

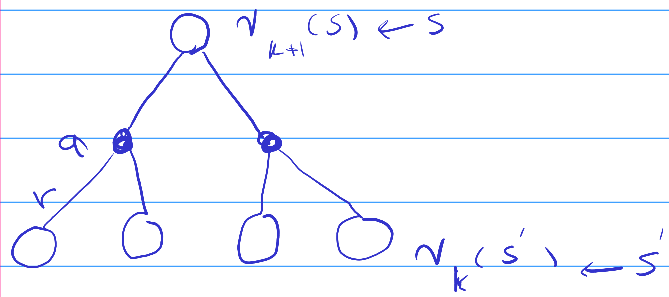
using synchronous backups:

at each iteration $k+1$:

for all states $s \in S$:

update $v_{k+1}(s)$ from $v_k(s')$

where s' is the successor state



$$v_{k+1}(s) = \sum_{a \in A} \pi(a|s)\left(R_s^a + \gamma \sum_{s' \in S} P_{ss'}^a v_k(s')\right)$$

$$v^{k+1} = R^\pi + \gamma R^\pi v^k$$

[gridworld example great]

policy evaluation: estimate v_π, iterative policy evaluation

Policy Iteration

given a policy

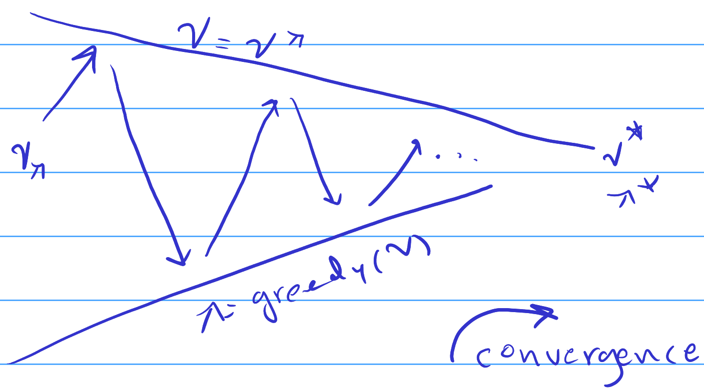
→ evaluate the policy π $\left[v_\pi(s) = \mathbb{E}\left[R_{t+1} + \gamma R_{t+2} + \dots \mid S_t = s\right] \right]$

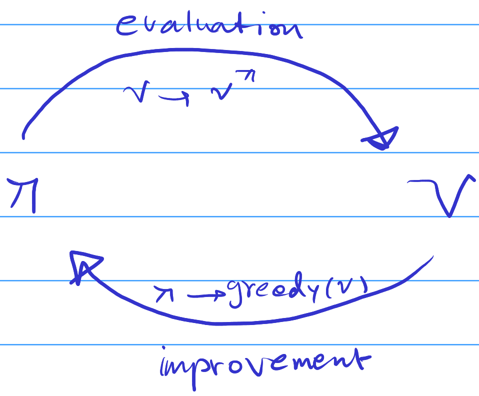
→ Improve the policy by acting greedily with v_π

$\pi' = \text{greedy}(v_\pi)$

Policy improvement: Generate $\pi' > \pi$

greedy policy improvement





Convergence?

considering a deterministic policy $a = \pi(s)$
we can improve the policy by acting greedily $\pi'(s) = \underset{a \in A}{\operatorname{argmax}}\, q_\pi(s,a)$
which improves the value from any step s over one step

$$q_\pi(s, \pi'(s)) = \max_{a \in A} q_\pi(s,a) \geq q_\pi(s, \pi(s)) = v_\pi(s)$$

and therefore improves the value function $v_{\pi'}(s) \geq v_\pi(s)$

$$\begin{aligned} v_\pi(s) &\leq q_\pi(s, \pi'(s)) = \mathbb{E}_{\pi'}[R_{t+1} + \gamma v_\pi(S_{t+1}) \mid S_t = s] \\ &\leq \mathbb{E}_{\pi'}[R_{t+1} + \gamma q_\pi(S_{t+1}, \pi'(S_{t+1})) \mid S_t = s] \\ &\leq \mathbb{E}_{\pi'}[R_{t+1} + \gamma R_{t+2} + \dots \mid S_t = s] = v_{\pi'}(s) \end{aligned}$$

If improvement stops,

$$q_\pi(s, \pi'(s)) = \max_{a \in A} q_\pi(s,a) = q_\pi(s, \pi(s)) = v_\pi(s)$$

then Bellman optimality equation becomes satisfied

$$v_\pi(s) = \max_{a \in A} q_\pi(s,a)$$

Principle of Optimality

a policy $\pi(a|s)$ achieves optimal value from state s $v_\pi(s) = v_*(s)$ iff
for any s' reachable from s,
π achieves the optimal value from s', $v_\pi(s') = v_*(s')$

→ at each state, the optimal policy includes
taking the best action +
following the optimal policy from the next state

Value Iteration (1)

if we know the solution to subproblem $v_*(s')$

then $$v_*(s) \leftarrow \max_{a \in A} R_s^a + \gamma \sum_{s' \in S} P_{ss'}^a v_*(s')$$

(one-step look ahead)

Basically start with final rewards and work backwards

$1 \cdots \sim \cdots \Leftarrow G$

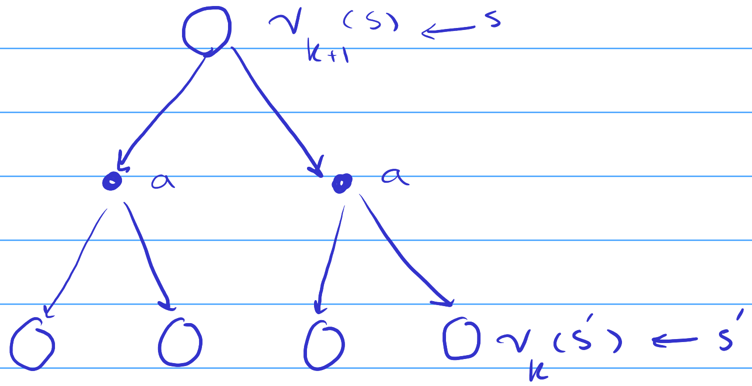
in practice goal state may not be obvious, but
it works by state sweep over entire st space

Value Iteration (2)

$v_1 \to v_2 \to \dots \to v_*$

at each iteration $k+1$
  for all states $s \in S$
    update $v_{k+1}(s)$ from $v_k(s')$

* intermediate value function may not correspond to any policy



$$v_{k+1}(s) = \max_{a \in A} \left(R_s^a + \gamma \sum_{s' \in S} P_{ss'}^a v_k(s') \right)$$

$$v_{k+1} = \max_{a \in A} R^a + \gamma P^a v_k$$

Problem	equation	algorithm
prediction	Bellman Expectation	iter policy eval
control	BE + Greedy policy impr	policy iteration
control	Bellman Optimality	Value iteration