Reinforcement Learning

- Sarath Chandar

4. Dynamic Bogramming

4. Dynamic Programming

- Collection of algorithms that can be used to compute optimal policies given a perfect model of the environment as a Markov decision Process (MDP).

Limitations :- 1. Assumes a perfect model.

2. Computationally expensive.

However, it provides essential foundation for the understanding of the future algorithms.

Assume that the environment is a finite MDP.

i.e. State S, action A and reward R are finite, the dynamics are given by

$$p(s', r \mid s, a) \quad \forall\, s \in S, a \in A(s), r \in R$$

and $s' \in S^+$ (where S^+ is S plus a terminal state for episodic setting)

Key idea: use of value functions to organize and structure the search for good policies.

Optimal value fns V_*, q_* satisfy the Bellman optimality equations.

$$V_*(s) = \max_a \mathbb{E}\left[R_{t+1} + \gamma V_*(S_{t+1}) \,\middle|\, S_t = s, A_t = a \right]$$

$$= \max_a \sum_{s', r} p(s', r | s, a) \left[r + \gamma V_*(s') \right]$$

$$q_*(s,a) = \mathbb{E}\left[R_{t+1} + \gamma \max_{a'} q_*(S_{t+1}, a') \,\middle|\, S_t = s, A_t = a \right]$$

$$= \sum_{s', r} p(s', r | s, a) \left[r + \gamma \max_{a'} q_*(s', a') \right]$$

DP algorithms are obtained by turning Bellman equations into update rules for improving approximations of the desired value functions.

Policy Evaluation (Prediction):

How to compute the state value fn V_π for an arbitrary policy π?

$$V_\pi(s) = \mathbb{E}\left[G_t \,|\, S_t = s \right]$$

$$= \mathbb{E}_\pi \left[R_{t+1} + \gamma G_{t+1} \mid S_t = s \right]$$

$$= \mathbb{E}_\pi \left[R_{t+1} + \gamma V_\pi(S_{t+1}) \mid S_t = s \right]$$

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

where $\pi(a|s)$ = prob. of taking action 'a' in state 's' under policy π.

The existence and uniqueness of V_π are guaranteed as long as either $\gamma < 1$ or eventual termination is guaranteed from all states under policy π.

Note: If the env's dynamics are completely known, then $(\star)$ is a system of $|S|$ simultaneous linear equations in $|S|$ unknowns. The solution is straight forward but tedious.

Iterative soln:- Consider a sequence of approximate value fns $v_0, v_1, v_2 \ldots$ each mapping S^+ to $\mathbb{R}$.

The initial approx. is chosen arbitrarily (except that the terminal state, if any, must be given value 0).

Each successive approx. is obtained by using the Bellman equation for V_π as an update rule.

$$V_{k+1}(s) = \mathbb{E}\left[R_{t+1} + \gamma V_k(S_{t+1}) \mid S_t = s \right]$$

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

$\forall s \in \mathcal{S}$.

① $V_k = V_\pi$ is a fixed point for this update rule.

② the sequence $\{V_k\}$ can be shown to converge to V_π as $k \to \infty$ under the same conditions that guarantee the existence of V_π.

This algorithm is known as iterative policy evaluation.

These updates are expected updates because they are based on an expectation over all possible next states rather than on a sample next state.

Note 1: One could use two arrays, one for old values $V_k(s)$ and one for new values $V_{k+1}(s)$. Or one can also do in-place updates which converges faster.

Note 2: One round of update of all the states is called as 'sweep' through the state space.

Note 3: Iterative policy evaluation only converges in limit. However, in practice we stop when the updates are sufficiently small.

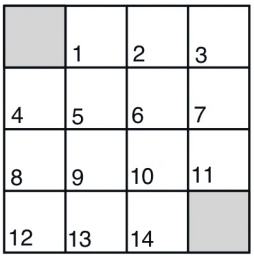
Iterative Policy Evaluation, for estimating $V \approx v_\pi$

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 $\Delta < \theta$

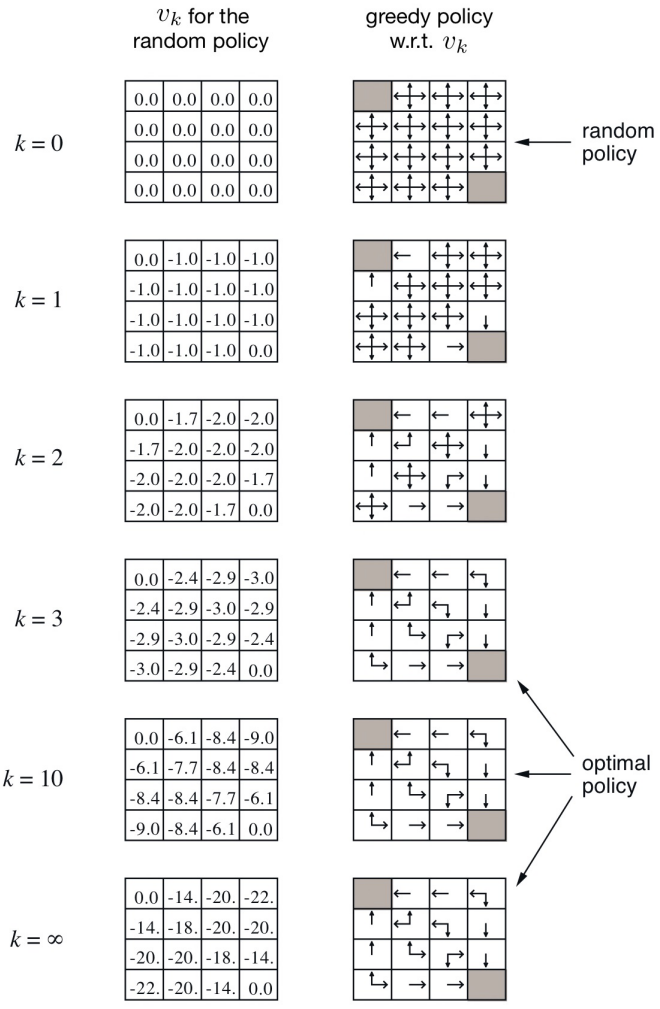
Example:

actions



$R_t = -1$
on all transitions

undiscounted, episodic setting.



Policy improvement:-

One reason for computing the value fn for a policy is to help find better policies.

Given V_π for some arbitrary policy π,

For some state s, we would like to know whether or not we should change the policy to deterministically choose an action $a \neq \pi(s)$.

Consider selecting 'a' in 's' and thereafter following the existing policy π. The value of this way of behaving is

$$q_\pi(s,a) = \mathbb{E}\left[R_{t+1} + \gamma v_\pi(S_{t+1}) \mid S_t = s, A_t = a\right]$$

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

If this is greater than $V_\pi(s)$, then the new policy π' which takes action 'a' in state 's' and follow π for rest of the states would be a better policy.

Policy improvement theorem:- (PIT)

Let π and π' be an pair of deterministic policies such that, for all $s \in S$

$$q_\pi(s, \pi'(s)) \geq v_\pi(s)$$

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

i.e. $$v_{\pi'}(s) \geq v_\pi(s) \quad \forall s \in S.$$

Proof:

$$v_\pi(s) \leq q_\pi(s, \pi'(s))$$

$$= \mathbb{E}\left[R_{t+1} + \gamma v_\pi(S_{t+1}) \mid S_t = s, A_t = \pi'(s) \right]$$

$$= \mathbb{E}_{\pi'}\left[R_{t+1} + \gamma v_\pi(S_{t+1}) \mid S_t = s \right]$$

$$\leq \mathbb{E}_{\pi'}\left[R_{t+1} + \gamma q_\pi(S_{t+1}, \pi'(S_{t+1})) \mid S_t = s \right]$$

$$= \mathbb{E}_{\pi'}\left[R_{t+1} + \gamma \mathbb{E}_{\pi'}\left[R_{t+2} + \gamma v_\pi(S_{t+2}) \mid S_{t+1}, A_{t+1} = \pi'(S_{t+1}) \right] \mid S_t = s \right]$$

$$= \mathbb{E}_{\pi'}\left[R_{t+1} + \gamma R_{t+2} + \gamma^2 v_\pi(S_{t+1}) \mid S_t = s \right]$$

$$\leq \mathbb{E}_{\pi'}\left[R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+3} + \gamma^3 V_\pi(S_{t+3}) \mid S_t = s \right]$$

$$\vdots$$

$$\leq \mathbb{E}_{\pi'}\left[R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+3} + \gamma^3 R_{t+4} + \cdots \mid S_t = s \right]$$

$$= V_{\pi'}(s)$$

Given the value fn. for some Policy π, Consider the following greedy policy π' w.r.t the value fn.

$$\pi'(s) = \underset{a}{\operatorname{argmax}}\ q_\pi(s,a)$$

$$= \underset{a}{\operatorname{argmax}}\ \mathbb{E}\left[R_{t+1} + \gamma V_\pi(S_{t+1}) \mid S_t = s, A_t = a \right]$$

$$= \underset{a}{\operatorname{argmax}} \sum_{s',r} p(s', r \mid s, a)\left[r + \gamma V_\pi(s') \right]$$

By PIT, this greedy policy should be as good as or better than the original policy.

The process of making a new policy that improves on an original policy, by making it greedy w.r.t. the value fn of the original policy is called Policy improvement.

Suppose the new greedy policy π' is as good as, but not better than π, then

$$v_\pi = v_{\pi'}.$$

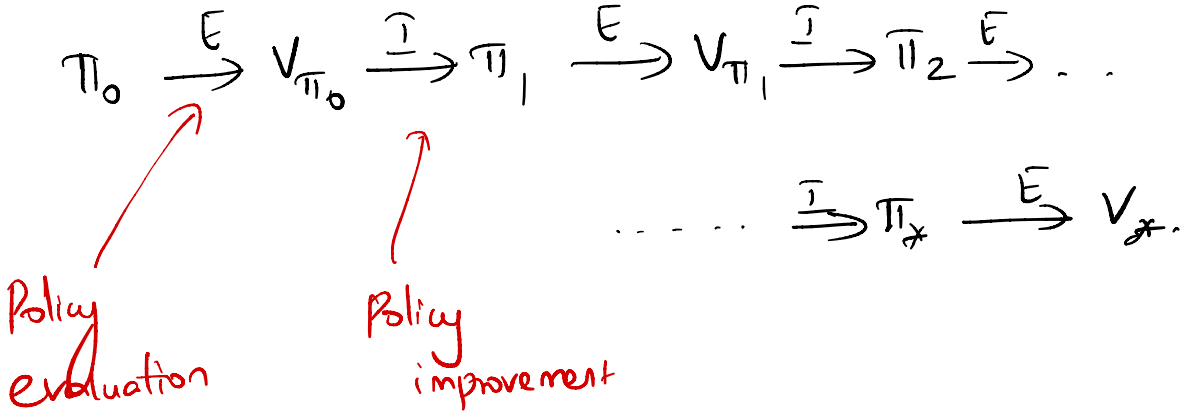
$$v_{\pi'}(s) = \max_a \mathbb{E}\left[R_{t+1} + \gamma v_{\pi'}(s_{t+1}) \mid S_t = s, A_t = a\right]$$

This is same as Bellman Optimality eqn.

Thus $v_{\pi'} = v_*$. and both π, π' should be optimal policies.

Policy Iteration: -

Sequence of monotonically improving policies and value fns.



Each policy is guaranteed to be a strict improvement over the previous one (unless it is already optimal).

Because a finite MDP has only a finite number of policies, this process must converge to an optimal policy and optimal value fn in a finite number of iterations.

Policy Iteration (using iterative policy evaluation) for estimating $\pi \approx \pi_*$

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 $s \in \mathcal{S}$:
   $v \leftarrow V(s)$
   $V(s) \leftarrow \sum_{s',r} p(s', r|s, \pi(s))[r + \gamma V(s')]$
   $\Delta \leftarrow \max(\Delta, |v - V(s)|)$
   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 $\leftarrow \pi(s)$
   $\pi(s) \leftarrow \arg\max_a \sum_{s',r} p(s', r|s, a)[r + \gamma V(s')]$
   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 :- (VI)

Limitation of policy iteration :- Each of its iteration involves policy evaluation, which itself is an iterative computation. Howevery, Policy evaluation only converges in limit. When to truncate policy evaluation step?

Value iteration stops policy evaluation after just one sweep.

The update rule for value iteration combines the policy improvement and truncated policy evaluation steps.

$$v_{k+1}(s) = \max_a \mathbb{E}\left[R_{t+1} + \gamma v_k(S_{t+1}) \mid S_t = s, A_t = a\right]$$

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

$\forall s \in S$

For arbitrary v_0, the sequence $\{v_k\}$ can be shown to converge to v_* under same conditions

that guarantee the existence of v_*.

Note 1: VI is obtained by simply turning the Bellman optimality eqn to an update rule.

Note 2: VI is identical to policy evaluation except that it requires max be taken over all actions.

Value Iteration, for estimating $\pi \approx \pi_*$

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

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

Many variations of interleaving policy evaluation with policy improvement is possible!

Asynchronous DP:-

DP requires sweep over entire state space. This is not possible in many problems.

Ex: Backgammon has ove 10^{20} states!

Asynchronous DP → in-place DP algorithms with no systematic sweeping.

Just update whatever state is next available. This makes it easier to intermix computation with real-time interaction. To solve a given MDP, we can run an iterative DP algorithm at the same time that an agent is actually experiencing the MDP.

Agent's experience → used to decide what state to update.

Latest value & Policy info from DP } → can guide the agent's decision making.

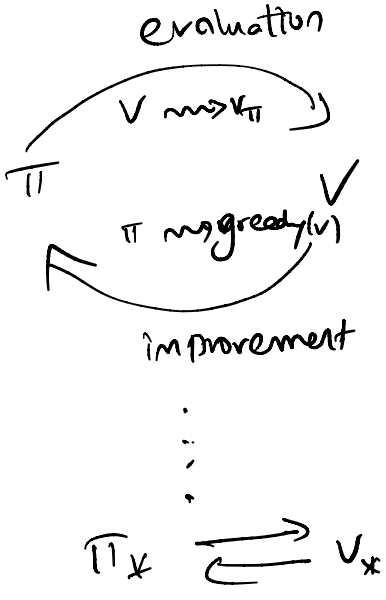
Generalized Policy Iteration:- (GPI)

Policy iteration consists of two simultaneous, interacting processes

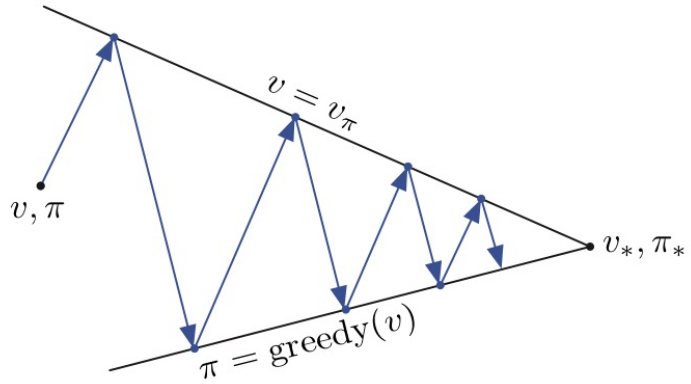
→ one making the value fn consistent with the current policy (Policy evaluation)

→ the other making the policy greedy w.r.t. the current value fn (Policy improvement)

GPI - idea of letting policy evaluation and policy improvement interact, independent of the granularity and other details of two processes.



These two processes are both Competing and Cooperating.



Efficiency of DP:-

DP finds an optimal policy in time that is polynomial in the number of states and actions.

n = # of states

k = # of actions.

Total # of deterministic policies = k^n.

DP is exponentially faster than direct search!

Linear programming can be used to solve MDP but they are not as scalable as DP.
