Reinforcement Learning Equations

Aaron Hao Tan

Fininte Markov Decision Processes
Components of MDP: Decision epochs, State, Action, Transition probability, Rewards

$$\{T, S, A_s, p_t(\cdot|s,a), r_t(s,a)\} \tag{1}$$

A state is *Markov* if and only if

$$\mathbb{P}\left[S_{t+1}|S_t\right] = \mathbb{P}\left[S_{t+1}|S_1,\ldots,S_t\right] \tag{2}$$

State-transition probabilities (p characterize the environment's dynamics). More often used than equations 4 and 5.

$$\begin{aligned} p(s'|s,a) &= Pr[S_t = s'|S_{t-1} = s, A_{t-1} = a] \\ &= \sum_{r\in\mathcal{R}} p(s',r|s,a) \end{aligned} \tag{3}$$

Expected rewards for state-action pairs and for state-action-next-state triples. The current reward depends on the previous state and action.

$$r(s,a) \doteq \mathbb{E}\left[R_t|S_{t-1}=s, A_{t-1}=a\right] = \sum_{r\in\mathcal{R}} r \sum_{s'\in\mathcal{S}} p\left(s',r|s,a\right) \tag{4}$$

$$r\left(s,a,s'\right) \doteq \mathbb{E}\left[R_t|S_{t-1}=s, A_{t-1}=a, S_t=s'\right] = \sum_{r\in\mathcal{R}} r\frac{p\left(s',r|s,a\right)}{p\left(s'|s,a\right)} \tag{5}$$

Policy
- Probabilistic is inferior to deterministic

$$\pi(a|s) = Pr(A_t = a|S_t = s) \tag{6}$$

Returns (sum of rewards)

$$G_t \doteq R_{t+1} + R_{t+2} + R_{t+3} + \cdots + R_T \tag{7}$$

Discounted Returns (γ = discount rate). If $\gamma < 1$, the infinite sum in equation 8 would have a finite value. If $\gamma = 0$, the agent is concerned with only maximizing immediate rewards (myopic). If the reward is +1, the return is equivalent to $G_t = \sum_{k=0}^{\infty}\gamma^k = \frac{1}{1-\gamma}$.

$$G_t \doteq R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+3} + \cdots = \sum_{k=0}^{\infty}\gamma^k R_{t+k+1} \tag{8}$$

$$\begin{aligned} G_t &\doteq R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+3} + \gamma^3 R_{t+4} + \cdots \\ &= R_{t+1} + \gamma\left(R_{t+2} + \gamma R_{t+3} + \gamma^2 R_{t+4} + \cdots\right) \\ &= R_{t+1} + \gamma G_{t+1} \end{aligned} \tag{9}$$

State Value Function. Value of terminal state (if any) is always zero.

$$v_\pi(s) \doteq \mathbb{E}_\pi[G_t|S_t = s] = \mathbb{E}_\pi[\sum_{k=0}^{\infty}\gamma^k R_{t+k+1}|S_t = s] \quad \forall s\in S \tag{10}$$

Bellman Equation for v_π: the value of the start state must equal the (discounted) value of the expected next state, plus the reward expected along the way. Assumes discrete and is invariant of time (time doesn't matter).

$$\begin{aligned} v_\pi(s) &= E_\pi[R_{t+1} + \gamma\cdot G_{t+1}|S_t = s] \\ &= \sum_a \pi(a|s)\sum_{s'}\sum_r p(s',r|s,a)[r + \gamma\mathbb{E}_\pi[G_{t+1}|S_{t+1} = s']] \\ &= \sum_a \pi(a|s)\sum_{s',r} p\left(s',r|s,a\right)\left[r + \gamma v_\pi\left(s'\right)\right] \quad \forall s\in S \end{aligned} \tag{11}$$

Action Value Function

$$q_\pi(s,a) \doteq \mathbb{E}_\pi\left[G_t|S_t=s, A_t=a\right] = \mathbb{E}_\pi\left[\sum_{k=0}^{\infty}\gamma^k R_{t+k+1}|S_t=s, A_t=a\right] \tag{12}$$

Bellman Equation for $q_\pi(s,a)$

$$q_\pi(s,a) = \sum_{s',r} p\left(s',r|s,a\right)\left[r+\gamma\sum_{a'}\pi\left(a'|s'\right)q_\pi\left(s',a'\right)\right] \tag{13}$$

The following figure represents the state value and action state value functions graphically.

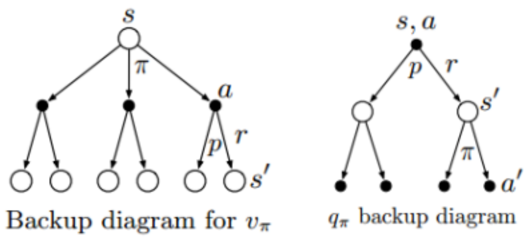


Figure 1: Value and State-Value Functions

Optimal State Value Function. A policy π is better or equal to another policy π' if its expected return is greater than or equal to that of π' for all states ($\pi' \geq \pi'$ if and only if $v_\pi(s) \geq v_{\pi'}(s)$).

$$v_*(s) \doteq \max_\pi v_\pi(s) \tag{14}$$

Optimal Action Value Function and its relation to v_*. Equation 15 gives the expected return for taking action a in state s and thereafter following an optimal policy. Comparing with equation 11, $v_\pi(s)$ represents the value of a state as the summation of future returns. In equation 15, $v_*\left(S_{t+1}\right)$ gives the optimal value of all future states from the given state and action.

$$\begin{aligned} q_*(s,a) &\doteq \max_\pi q_\pi(s,a) \\ &= \mathbb{E}\left[R_{t+1}+\gamma v_*\left(S_{t+1}\right)|S_t=s, A_t=a\right] \end{aligned} \tag{15}$$

Relationship between q_* and v_*.
- The value of a state under an optimal policy must equal the expected return for the best action from that state.
- Second last line of equation 16 shows that $G_t = R_{t+1}+\gamma v_*(S_{t+1})$ when following π_*.
- The last line is the same as equation 17. For finite MDPs, the last line of equation 16 has a unique solution.
- R_{t+1} has nothing to do with policy because the first action was not taken with policy
- max operator turns equation 16 in to a system of nonlinear equations

$$\begin{aligned} v_*(s) &= \max_{a\in A(s)} q_*(s,a) \\ &= \max_a q_*(s,a) \\ &= \max_a \mathbb{E}_{\pi_*}[G_t|S_t=s, A_t=a] \\ &= \max_a \mathbb{E}[R_{t+1}+\gamma v_*(S_{t+1})|S_t=s, A_t=a] \\ &= \max_a \sum_{s',r} p(s',r|s,a)[r+\gamma v_*(s')] \end{aligned} \tag{16}$$

Bellman Optimality Equations

$$\begin{aligned} v_*(s) &= \max_a \mathbb{E}\left[R_{t+1}+\gamma v_*\left(S_{t+1}\right)|S_t=s, A_t=a\right] \\ &= \max_a \sum_{s',r} p\left(s',r|s,a\right)\left[r+\gamma v_*\left(s'\right)\right] \end{aligned} \tag{17}$$

$$\begin{aligned} q_*(s,a) &= \mathbb{E}\left[R_{t+1}+\gamma \max_{a'} q_*\left(S_{t+1},a'\right)|S_t=s, A_t=a\right] \\ &= \sum_{s',r} p\left(s',r|s,a\right)\left[r+\gamma\max_{a'} q_*\left(s',a'\right)\right] \end{aligned} \tag{18}$$

In v_*, the next best action is selected based on the expected reward and value of future states. In q_*, the state and action is given as well as the reward. From the new state s', the best action is chosen.

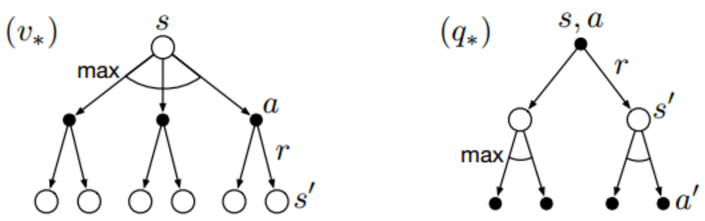


Figure 2: Bellman Optimality Equations

Dynamic Programming
Policy Evaluation
- turn Bellman equation from equation 11 in to an update rule

$$\begin{aligned} v_{k+1}(s) &\doteq \mathbb{E}_\pi \left[R_{t+1} + \gamma v_k\left(S_{t+1}\right) | S_t = s\right] \\ &= \sum_a \pi(a|s) \sum_{s',r} p\left(s', r|s, a\right)\left[r + \gamma v_k\left(s'\right)\right] \end{aligned} \tag{19}$$

Policy Evaluation Steps
- for state in states
—- for action in actions available in each state
——— calculate value of state using equation 19
——— (value = sum all values of each action together for each state (ie. 4 actions in a state = 1 total value))
—- replace old value of state with newly calculated value
—- calculate the change in value of each state with only the maximum difference remembered
- stop looping if the maximum change in state values falls below a threshold

Policy Improvement
- greedy policy = guaranteed to be optimal
- one-step greedy lookahead

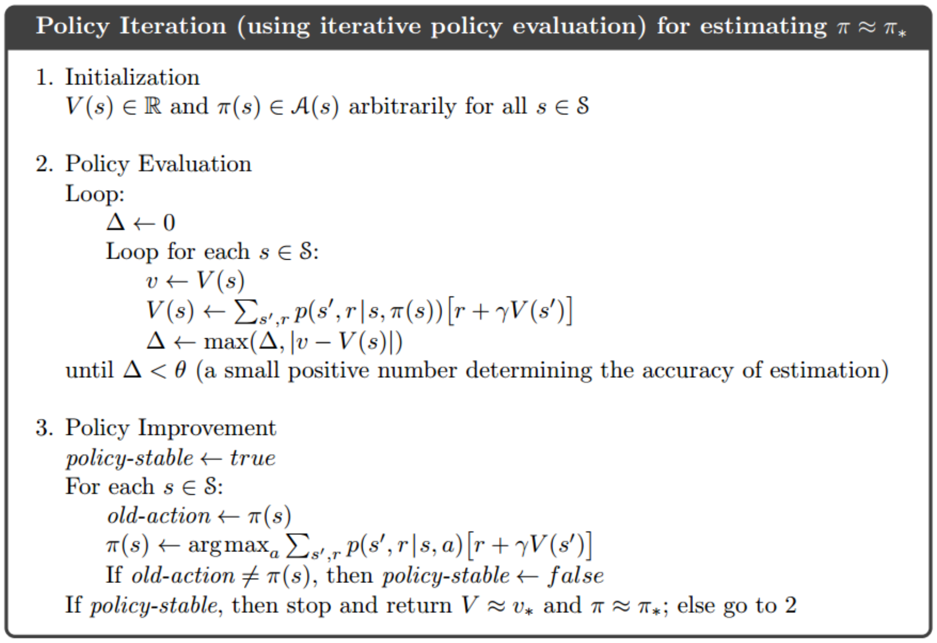
$$\begin{aligned} \pi'(s) &\doteq \arg\max_a q_\pi(s, a) \\ &= \arg\max_a \mathbb{E}\left[R_{t+1} + \gamma v_\pi\left(S_{t+1}\right) | S_t = s, A_t = a\right] \\ &= \arg\max_a \sum_{s',r} p\left(s', r|s, a\right)\left[r + \gamma v_\pi\left(s'\right)\right] \end{aligned} \tag{20}$$

Policy is represented $s \times a$ table where each row represents a state and each column represents an action. The value within each entry of the table represent the probability of taking that action, when in that state, using a probabilistic policy.

At this stage, there is a value calculated for every state using policy evaluation already.

Policy Improvement Steps
- for state in states
—- choose the best action with the current policy (argmax)
—- for action in actions available in each state
——— calculate action values for each action using equation 20 without the argmax
———(this creates a $1 \times a$ vector where each element is an action value q)
—- select the best action in the $1 \times a$ vector (equation 20 with argmax)
—- if action chosen with the current policy is not the calculated best action
——— policy is unstable
—- update the policy with the new best action

Policy Iteration
- built on the fundamentals of value iteration
- more efficient (CPU time) than value iteration



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 $s \in \mathcal{S}$:
           $v \leftarrow V(s)$
           $V(s) \leftarrow \sum_{s',r} p(s', r \mid s, \pi(s)) \big[r + \gamma V(s')\big]$
           $\Delta \leftarrow \max(\Delta, |v - V(s)|)$
   until $\Delta < \theta$ (a small positive number determining the accuracy of estimation)

3. Policy Improvement
   $policy\text{-}stable \leftarrow true$
   For each $s \in \mathcal{S}$:
       $old\text{-}action \leftarrow \pi(s)$
       $\pi(s) \leftarrow \arg\max_a \sum_{s',r} p(s', r \mid s, a) \big[r + \gamma V(s')\big]$
       If $old\text{-}action \neq \pi(s)$, then $policy\text{-}stable \leftarrow false$
   If $policy\text{-}stable$, then stop and return $V \approx v_*$ and $\pi \approx \pi_*$; else go to 2
```

Figure 3: Policy Iteration

Value Iteration
- Policy iteration is faster because policy converges quicker than value functions
- Bellman Optimality Equation (equation 17) turned in to an update rule
- Formally requires infinite number of iterations to converge exactly to v_* but we stop before then
- Only guarantees $\epsilon - optimality$ (theoretically)
- All algorithms converge to an optimal policy for discounted finite MDPs
- When γ is close to 1, iteration goes forever = making value iteration inefficient
- Policy iteration converges faster even for smaller γ - Main difference between Value Iteration and Policy Iteration
– In value iteration, instead of summing values from all actions per state, take only the best action value per state as the value of that state
- Value iteration update is identical to policy evaluation except that the max is taken over all actions

$$\begin{aligned} v_{k+1}(s) &\doteq \max_a \mathbb{E}\left[R_{t+1} + \gamma v_k\left(S_{t+1}\right) | S_t = s, A_t = a\right] \\ &= \max_a \sum_{s',r} p\left(s', r | s, a\right)\left[r + \gamma v_k\left(s'\right)\right] \end{aligned} \tag{21}$$

Value Iteration Steps
- for state in states
—- for action in actions available in each state
——— calculate action values for each action using equation 21 without the max
———(this creates a $1 \times a$ vector where each element is an action value q)
—- value = the best action value in the $1 \times a$ vector (equation 21 with max)
—- calculate the change in value of each state with only the maximum difference remembered
- stop looping if the maximum change in state values falls below a threshold
- for state in states
—- for action in actions available in each state
——— calculate action values for each action using equation 21 without the max
———(this creates a $1 \times a$ vector where each element is an action value q)
—- select the best action in the $1 \times a$ vector (equation 21 with argmax)
—- overwrite policy table with best action

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

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

Figure 4: Value Iteration