Examples using Policy and Value Functions

(from *Sutton and Barto*)
Compiled by D. Gueorguiev 1/21/2024

Table of Contents

5x5 Gridworld with special states

The cells of the grid correspond to the states of the environment. At each cell, four actions are possible: ***north***, ***south***, ***east***, and ***west*** which deterministically cause the agent to move one cell in the respective direction on the grid. Actions that would take the agent off the grid leave its position unchanged, but also result in a reward of -1. Other actions result in a reward of 0, except those that move the agent out of the special states A and B. From state A, all four actions yield a reward of $+10$ and take the agent to A'. From state B, all actions yield a reward of $+5$ and take the agent to B'.

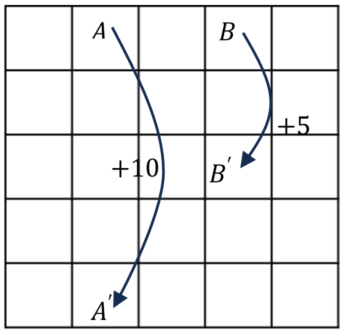

Figure 1: 5x5 `Gridworld` Example with special states

Suppose the agent selects all four actions with equal probability in all states. Figure 2 below shows the value function, v_π , for this policy, for the discounted reward case of $\gamma = 0.9$.

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

Figure 2: State-value function for `Gridworld`

Now let us denote with $s_i, i = 1..25$ and we set $p = l/5\,, q = l\,\%\,5$.
Then s_i represents the state of the position $(p,q), p,q \in [1,5]$ in the grid. Thus $\mathcal{S} \equiv [1,25]$.

Finding state value and action value functions from the Bellman's equations

Let us assume random (equiprobable) policy $\pi^r(a|s_i)$ so that all $a \in [north, south, east, west]$ are equally probable with probability $^1/_4$. The reward and the new state for every pair of action and state as a tuple (r, s') is given with the following lookup table $\tau^\ell(s,a)$:

action\state	s_1	s_2	s_3	s_4	s_5	s_6	s_7	s_8	s_9	s_{10}
$north$	$(-1,s_1)$	$(10,s_{22})$	$(-1,s_3)$	$(5,s_{14})$	$(-1,s_5)$	$(0,s_1)$	$(0,s_2)$	$(0,s_3)$	$(0,s_4)$	$(0,s_4)$
$south$	$(0,s_6)$	$(10,s_{22})$	$(0,s_8)$	$(5,s_{14})$	$(0,s_{10})$	$(0,s_{11})$	$(0,s_{12})$	$(0,s_{13})$	$(0,s_{14})$	$(0,s_{15})$
$east$	$(0,s_2)$	$(10,s_{22})$	$(0,s_4)$	$(5,s_{14})$	$(-1,s_5)$	$(0,s_7)$	$(0,s_8)$	$(0,s_9)$	$(0,s_{10})$	$(-1,s_{10})$
$west$	$(-1,s_1)$	$(10,s_{22})$	$(0,s_2)$	$(5,s_{14})$	$(0,s_4)$	$(-1,s_6)$	$(0,s_6)$	$(0,s_7)$	$(0,s_8)$	$(0,s_9)$

action\state	s_{11}	s_{12}	s_{13}	s_{14}	s_{15}	s_{16}	s_{17}	s_{18}	s_{19}	s_{20}
$north$	$(0,s_6)$	$(0,s_7)$	$(0,s_8)$	$(0,s_9)$	$(0,s_{10})$	$(0,s_{11})$	$(0,s_{12})$	$(0,s_{13})$	$(0,s_{14})$	$(0,s_{15})$
$south$	$(0,s_{16})$	$(0,s_{17})$	$(0,s_{18})$	$(0,s_{19})$	$(0,s_{20})$	$(0,s_{21})$	$(0,s_{22})$	$(0,s_{23})$	$(0,s_{24})$	$(0,s_{25})$
$east$	$(0,s_{12})$	$(0,s_{13})$	$(0,s_{14})$	$(0,s_{15})$	$(-1,s_{15})$	$(0,s_{17})$	$(0,s_{18})$	$(0,s_{19})$	$(0,s_{20})$	$(-1,s_{20})$
$west$	$(-1,s_{11})$	$(0,s_{11})$	$(0,s_{12})$	$(0,s_{13})$	$(0,s_{14})$	$(-1,s_{16})$	$(0,s_{16})$	$(0,s_{17})$	$(0,s_{18})$	$(0,s_{19})$

action\state	s_{21}	s_{22}	s_{23}	s_{24}	s_{25}
$north$	$(0,s_{16})$	$(0,s_{17})$	$(0,s_{18})$	$(0,s_{19})$	$(0,s_{20})$
$south$	$(-1,s_{21})$	$(-1,s_{22})$	$(-1,s_{23})$	$(-1,s_{24})$	$(-1,s_{25})$
$east$	$(0,s_{22})$	$(0,s_{23})$	$(0,s_{24})$	$(0,s_{25})$	$(-1,s_{25})$
$west$	$(-1,s_{21})$	$(0,s_{21})$	$(0,s_{22})$	$(0,s_{23})$	$(0,s_{24})$

Table 1: a tuple containing the reward r and new state s' when taking an action a from state s.

The following pseudo code will do that computation:

```
# define the action set of Gridworld
from enum import Enum
class Action(Enum):
    North=1
    South=2
    East=3
    West=4

# generate the lookup table shown on Table 1
lookup_table_rsprime = dict()
for r in range(0,5):  # current row
    for c in range(0,5):  # current column
        for a in Action:
            if (r,c) == (0,1):
                lookup_table_rsprime[(5*r+c,a)] = (10, (4,1))
            elif (r,c) == (0,3):
                lookup_table_rsprime[(5*r+c,a)] = (5, (2,3))
            else:
                if a == Action.North and r == 0 or \
                   a == Action.South and r == 4 or \
                   a == Action.East and c == 4 or \
                   a == Action.West and c == 0:
                    lookup_table_rsprime[(5*r+c,a)] = (-1, (r,c))
                elif a == Action.North:
                    lookup_table_rsprime[(5*r+c,a)] = (0, (r-1,c))
                elif a == Action.South:
                    lookup_table_rsprime[(5*r+c,a)] = (0, (r+1,c))
                elif a == Action.East:
                    lookup_table_rsprime[(5*r+c,a)] = (0, (r,c+1))
                elif a == Action.West:
             lookup_table_rsprime[(5*r+c,a)] = (0, (r,c-1))
```
Code Excerpt 1: implementation of the lookup table $\tau^{\ell}(s,a)$

Then the function of the MDP dynamics $p(s',r|s,a)$ is given with

$$p(s',r|s,a) = \Delta\Big((s',r),\ \tau^{\ell}(s,a)\Big)\ \forall s \in \{s_1,\dots,s_{25}\} \text{ and } a \in \{north, south, east, west\} \quad (1)$$

Here the function $\Delta(\tau_1,\tau_2)$: $(\mathcal{S}\times\mathbb{R})\times(\mathcal{S}\times\mathbb{R}) \to \{0,1\}$ is defined as having value 1 if $\tau_1 = \tau_2$ and 0 otherwise ($\tau_1 \neq \tau_2$).

The following pseudocode will compute the function $p(s',r|s,a)$ of the MDP dynamics

```
def delta_fun(tau_1, tau_2):
    if tau_1 == tau_2:
        return 1
    else:
        return 0


def state_id(state):
    return 5*state[0]+state[1]


def mdp_dynamics(state_prime, reward, state, action):
    if not (state_id(state), action) in lookup_table_rsprime:
        raise ValueError(f"Unrecognized state and action pair! state: {state}, action: {action}")
    return delta_fun((reward, state_prime), lookup_table_rsprime[(state_id(state), action)])
```
Code Excerpt 2: Implementation of the function of the MDP dynamics $p(s',r|s,a)$

Let us compute v_π given the random (equiprobable) policy $\pi^{r}(a|s_i)$ specified in the previous paragraph.

We use the derived in the Appendix expression for (A.18)-(A.24):

$$p_\pi(s_j,r|s_i) \doteq \mathbb{E}[\pi|S_{t-1}=s_i] = \textstyle\sum_a p(s_j,r|s_i,a)\cdot\pi(a|s_i) \quad \text{(A.18)}$$

$$p_\pi(s_j|s_i) \doteq \textstyle\sum_r p_\pi(s_j,r|s_i) \quad \text{(A.19)}$$

$$r_\pi(s_j|s_i) \doteq \textstyle\sum_r p_\pi(s_j,r|s_i)\cdot r \quad \text{(A.20)}$$

For convenience we abbreviate:

$$p_{j,i} \doteq p_\pi(s_j|s_i)\ ,\ r_i \doteq \textstyle\sum_j r_\pi(s_j|s_i),\ S \doteq |\mathcal{S}| \qquad \text{(A.21)}$$

Using (A.16)-(A.21) in (3) leads to :

$$(1-\gamma p_{i,i})v_\pi(s_i) - \gamma\textstyle\sum_{j\neq i} p_{j,i}v_\pi(s_j) = r_i \qquad \text{(A.22)}$$

(A.22) represents a linear system of equations with respect to the $|\mathcal{S}|$ unknowns $v_\pi(s_i), s_i \in \mathcal{S}$.

Let us denote with $\boldsymbol{v_\pi}$ the column vector of the S unknowns $v_\pi(s_i), s_i \in \mathcal{S}$.
We denote with $\boldsymbol{A}$ the matrix formed by the elements and $\boldsymbol{b}$ the vector as shown below:

$$A = \begin{bmatrix} 1-\gamma p_{11} & p_{21} & \cdots & p_{S1} \\ p_{12} & 1-\gamma p_{22} & & p_{S2} \\ \vdots & & \ddots & \vdots \\ p_{1S} & p_{2S} & \cdots & 1-\gamma p_{SS} \end{bmatrix}, \quad b = \begin{bmatrix} r_1 \\ r_2 \\ \vdots \\ r_S \end{bmatrix} \qquad \text{(A.23)}$$

Then (A.22) in matrix form:

$$\boldsymbol{A} \cdot \boldsymbol{v_\pi} = \boldsymbol{b} \qquad \text{(A.24)}$$

First, we would like to compute $p_\pi(s_j, r|s_i)$ using (A.18):

$$p_\pi(s_j, r|s_i) \doteq \mathbb{E}[\pi|S_{t-1} = s_i] = \sum_a p(s_j, r|s_i, a) \cdot \pi(a|s_i) \quad \text{(A.18)}$$

$$p_{\pi^r}(s_j, r|s_i) = \sum_{a \in \mathcal{A}} \Delta\left((s_j, r), \tau^\ell(s_i, a)\right) \cdot \frac{1}{4} \qquad (2)$$

In pseudo-code we write:

```
def random_policy(action, state):
    return 1.0/len(Action)

def state_reward_distr(state_j, reward, state_i, policy):
    res = 0.0
    for action in Action:
        res += mdp_dynamics(state_j, reward, state_i, action)*policy(action, state_i)
    return res
```
Code Excerpt 3: implementation of $p_\pi(s_j, r|s_i)$

In the Gridworld problem we do not have more than one reward value when we transition from one state to another; This is true for any policy $\pi(a|s_i)$.

Thus (2) becomes

$$p_{\pi^r}(s_j|s_i) = \sum_{a \in \mathcal{A}} \Delta\left((s_j, r_{ji}), \tau^\ell(s_i, a)\right) \cdot \frac{1}{4} \qquad (3)$$

where r_{ji} denotes the reward for transitioning from s_i to s_j which is given with the function $r(s_j, s_i)$ defined with the following algorithm:

Given s_i and s_j find a such that s_j equals $\tau^\ell(s_i, a)[1]$
If such a does not exist return `-math.inf` for r_{ji}. Otherwise, return $\tau^\ell(s_i, a)[0]$

In pseudo-code this is expressed as:

```
def r(j, i):
    for a in Action:
        lookup_res = lookup_table_rsprime[(i, a)]
        if state_id(lookup_res[1]) == j:
            return lookup_res[0]
    return 0 # we assign zero for the reward of a missing state transition
```

Code Excerpt 4: Implementation of the state transition reward coefficients r_{ji}

Notice that (3) in fact is the quantity $p_\pi(s_j|s_i)$ defined in (A.19)

$p_\pi(s_j|s_i) \doteq \sum_r p_\pi(s_j, r|s_i)$ (A.19)

as the summation over r for given i and j always results in a single term which is entirely determined by specifying the indices i and j.

(3) computes the total probability to make a transition from state s_i to s_j given the equiprobable (random) policy π^r. As an illustration let us compute the probability $p_{\pi^r}(s_{22}|s_2)$:

$$p_{\pi^r}(s_{22}|s_2) = \frac{1}{4}\Big[\Delta\Big((s_{22}, 10), \tau^\ell(s_2, a_{north})\Big) + \Delta\Big((s_{22}, 10), \tau^\ell(s_2, a_{south})\Big) + \Delta\Big((s_{22}, 10), \tau^\ell(s_2, a_{east})\Big) + \Delta\Big((s_{22}, 10), \tau^\ell(s_2, a_{west})\Big)\Big] = 1$$

As another example let us compute $p_{\pi^r}(s_1|s_1)$:

$$p_{\pi^r}(s_1|s_1) = \tfrac{1}{4}\Big[\Delta\Big((s_1, -1), \tau^\ell(s_1, a_{north})\Big) + \Delta\Big((s_1, -1), \tau^\ell(s_1, a_{west})\Big)\Big] = \tfrac{1}{2}$$

For $p_{\pi^r}(s_{12}|s_{12})$:

Clearly, since no combination of action and reward from state s_{12} leads again into state s_{12} we conclude that $p_{\pi^r}(s_{12}|s_{12}) = 0$.

In pseudo-code this is expressed as:

```
def state_distr(state_j, state_i, policy):
    return state_reward_distr(state_j, r(state_id(state_j),state_id(state_i)), state_i, policy)
```

Code Excerpt 5: implementation of the distribution for transitioning from state s_i to s_j $p_\pi(s_j|s_i)$

Now using (A.20) we compute $r_\pi(s_j|s_i)$ which represents the expected reward for the transition from s_i to s_j

$r_\pi(s_j|s_i) \doteq \sum_r p_\pi(s_j, r|s_i) \cdot r$ (A.20)

In the Gridworld problem we do not have more than one reward value when we transition from one state to another; This is true for any policy $\pi(a|s_i)$. Hence:

$r_\pi(s_j|s_i) = p_\pi(s_j, r_{ji}|s_i) \cdot r_{ji}$ for any $\pi(a|s_i)$ (4)

Next, we compute the coefficient r_i given with (A.21)

$r_i \doteq \sum_j r_\pi(s_j|s_i)$ (A.21)

The expected reward coefficients $r_\pi(s_j|s_i)$ and the total reward r_i from state s_i are computed with the following pseudo-code:

```
def expected_reward_for_state_transition(state_j, state_i, policy):
    return state_distr(state_j, state_i, policy)*r(state_id(state_j), state_id(state_i))

def total_expected_reward_from_state(state, policy):
    res = 0
    for r in range(0,5):  # current row
        for c in range(0,5):  # current column
            res += expected_reward_for_state_transition((r,c), state, policy)
    return res
```

Code Excerpt 6: impl for the expected reward coefficients $r_\pi(s_j|s_i)$ and the total reward r_i from state s_i

Now we can solve the system of equations for $\boldsymbol{v_\pi}$ given with (A.23) and (A.24)

$$A = \begin{bmatrix} 1-\gamma p_{11} & p_{21} & \cdots & p_{S1} \\ p_{12} & 1-\gamma p_{22} & & p_{S2} \\ \vdots & & \ddots & \vdots \\ p_{1S} & p_{2S} & \cdots & 1-\gamma p_{SS} \end{bmatrix}, \quad b = \begin{bmatrix} r_1 \\ r_2 \\ \vdots \\ r_S \end{bmatrix} \qquad \text{(A.23)}$$

Then (A.22) in matrix form:

$$\boldsymbol{A} \cdot \boldsymbol{v_\pi} = \boldsymbol{b} \qquad \text{(A.24)}$$

//TODO: finish finding the state value and action value functions for Gridworld from Bellman's equations given the MDP dynamics probabilities

Iterative Policy Evaluation

We have

$$\begin{aligned} v_\pi(s) &= \sum_a \pi(a|s) \sum_{s'} \sum_r p(s', r|s,a) \left[r + \gamma \mathbb{E}_\pi[G_{t+1}|S_{t+1} = s']\right] \\ &= \sum_a \pi(a|s) \sum_{s'} \sum_r p(s', r|s,a) \left[r + \gamma v_\pi(s')\right] \quad \text{for all} \quad s \in \mathcal{S} \end{aligned} \qquad \text{(A.12)}$$

Let us compute

$$v_{k+1}(s) \doteq \mathbb{E}_\pi[R_{t+1} + \gamma v_k(S_{t+1})|S_t = s] = \sum_a \pi(a|s) \sum_{s',r} p(s', r|s,a)[r + \gamma v_k(s')] \;\; \forall s \in \mathcal{S}.$$

Clearly, $v_k = v_\pi$ is a fixed point in this update rule because the Bellman equation for v_π assures us of equality in this case.

//TODO: finish applying iterative policy evaluation in order to find state value and action value functions for the Gridworld example given equiprobable policy and the MDP dynamics probabilities

Policy Improvement

//TODO: finish policy improvement algorithm applied to the Gridworld example

4x4 Gridworld with terminal state

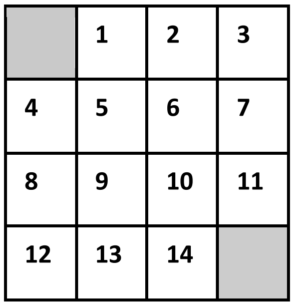

Figure 3: 4x4 `Gridworld` Example with terminal state. The terminal state is shown in gray

The non-terminal states are $\mathcal{S} = \{1,2,\ldots,14\}$. There are four actions possible in each state, $\mathcal{A} = \{\boldsymbol{up}, \boldsymbol{down}, \boldsymbol{right}, \boldsymbol{left}\}$ which deterministically cause the corresponding state transitions, except that actions that would take the agent off the grid in fact leave the state unchanged. This is an undiscounted, episodic task. The terminal state is shown in gray on Figure 3 above. The reward is -1 for all transitions until terminal state is reached. Thus, the expected reward function

$$r(s,a,s') = -1 \ \forall s, s' \in \mathcal{S}, \forall a \in \mathcal{A}.$$

For example here are few values of the function of the MDP dynamics of this Example:

$$p(6,-1|5,right) = 1 \ , p(7,-1|7,right) = 1 \ , p(10,r|5,right) = 0 \ \forall r \in \mathcal{R}$$

Iterative Policy Evaluation

We have

$$\begin{aligned} v_\pi(s) &= \textstyle\sum_a \pi(a|s) \sum_{s'} \sum_r p(s',r|s,a) \left[r + \gamma \mathbb{E}_\pi[G_{t+1}|S_{t+1} = s']\right] \\ &= \textstyle\sum_a \pi(a|s) \sum_{s'} \sum_r p(s',r|s,a) \left[r + \gamma v_\pi(s')\right] \quad \text{for all} \quad s \in \mathcal{S} \qquad \text{(A.12)} \end{aligned}$$

Let us compute

$$v_{k+1}(s) \doteq \mathbb{E}_\pi[R_{t+1} + \gamma v_k(S_{t+1})|S_t = s] = \textstyle\sum_a \pi(a|s) \sum_{s',r} p(s',r|s,a)[r + \gamma v_k(s')] \ \ \forall s \in \mathcal{S}.$$

Clearly, $v_k = v_\pi$ is a fixed point in this update rule because the Bellman equation for v_π assures us of equality in this case.

//TODO: finish applying iterative policy evaluation in order to find state value and action value functions for the Gridworld example given equiprobable policy and the MDP dynamics probabilities

Policy Improvement

//TODO: finish policy improvement algorithm applied to the 4x4 Gridworld example

Jack's Car Rental

Jack manages two locations for a nationwide car rental company. Each day, some number of customers arrive at each location to rent cars. If Jack has a car available, he rents it out and is credited \$10 by the national company. If he is out of cars at that location, then the business is lost. Cars become available for renting the day after they are returned. To help ensure that cars are available where they are needed, Jack can move them between the two locations overnight, at a cost of \$2 per car moved. We assume that the number of cars requested and returned at each location are Poisson random variables with probability for the respective number n equal to $\frac{\lambda^n}{n!}e^{-\lambda}$, where λ is the expected number. Suppose λ is 3 and 4 for rental requests at the first and the second locations and 3 and 2 for returns. That is, if we denote with n_i^+ the number of rental requests and with n_i^- the number of rental returns for location $i = 1,2$ we will have:

$$P(n_i^+) \sim \frac{\lambda_i^{+n_i^+}}{n_i^+!}e^{-\lambda_i^+}\ ;\ \ P(n_i^-) \sim \frac{\lambda_i^{-n_i^-}}{n_i^-!}e^{-\lambda_i^-}\ ; \lambda_1^+ = 3, \lambda_2^+ = 4, \lambda_1^- = 3, \lambda_2^- = 2 \qquad (1)$$

Let us denote with x_i the number of cars at location i. Obviously, we require $x_i \geq 0$.

We will impose the additional constraint:

$$x_i \leq 20, i = 1,2 \qquad (2)$$

That is, we require no more than 20 cars in each location. Any additional cars are returned to the nationwide company location which will be modeled with the surplus variable $y \geq 0$. Thus, instead of (2), we have:

$$x_i \leq 20, i = 1,2$$
$$x_i \geq 0,\ \ y \geq 0 \qquad (3)$$

The time is quantized in days hence all our variables will have the time indexed by day as a subscript. Thus, we rewrite (3) as:

$$x_{i,t} \leq 20, i = 1,2$$
$$x_{i,t} \geq 0,\ \ y_t \geq 0$$

if $x_{i,t} - n_i^+ + n_i^- \leq 20$
$x_{i,t+1} = x_{i,t} - n_i^+ + n_i^-$
else
$x_{i,t+1} = 20,\ y_{t+1} = x_{i,t} - n_i^+ + n_i^- - 20 \qquad (4)$

In MIP formulation we rewrite the constraints (4) as:

$$x_{i,t} - n_i^+ + n_i^- - Mz \leq 20$$
$$x_{i,t+1} + Mz \geq x_{i,t} - n_i^+ + n_i^-$$
$$x_{i,t+1} - Mz \leq x_{i,t} - n_i^+ + n_i^-$$

where $z \in \{0,1\}$ is a slack binary variable turning off and on a relevant set of constraints depending on the test condition evaluating to True or False.

Appendix

Notation and Definitions from Sutton and Barto's RL book

Distribution of the Dynamics of the MDP: defined through the following 4 arguments function:
$p(s', r|s, a) \doteq \Pr\{S_t = s', R_t = r|S_{t-1} = s, A_{t-1} = a\}$
which is the probability to get from state s to state s' with action a and with reward r.

Distribution of the state-transition probabilities: defined through the following 3 arguments function:
$p(s'|s, a) \doteq \Pr\{S_t = s'|S_{t-1} = s, A_{t-1} = a\} = \sum_{r\in\mathcal{R}} p(s', r|s, a)$

Markov Decision Process (abbrev *MDP*): a 5-tuple $(\mathcal{S}, \mathcal{A}, p, r, \gamma)$ with

- $\mathcal{S}$ is a set of states (finite or infinite, discrete, or continuous)
- $\mathcal{A}$ is a set of actions (finite or infinite, discrete, or continuous)
- $p(s', r|s, a) \doteq \Pr\{S_t = s', R_t = r|S_{t-1} = s, A_{t-1} = a\}$ is the function which describes the MDP dynamics i.e. probability to get from state s to state s' with action a and with reward r.
- $r : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$ defines a *reward function*
- $\gamma \in [0,1]$ is the discount factor which determines to what extent the focus is on the most recent rewards. with $\gamma = 1$ there is no focus on the most recent rewards only.

Note: There is another equivalent definition of Markov process which uses the *state-transition probabilities distribution* represented by the three-argument function $p : \mathcal{S} \times \mathcal{S} \times \mathcal{A} \to [0,1]$. With this definition the Markov Decision Process is defined as a 5-tuple $(\mathcal{S}, \mathcal{A}, p, r, \gamma)$, where:

- $\mathcal{S}$ is a set of states (finite or infinite, discrete, or continuous)
- $\mathcal{A}$ is a set of actions (finite or infinite, discrete, or continuous)
- $p(s'|s, a) \doteq \Pr\{S_t = s'|S_{t-1} = s, A_{t-1} = a\}$ is the distribution of the *state-transition probabilities* i.e. the probability to get from state s to state s' with action a.
- $r : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$ defines a *reward function*
- $\gamma \in [0,1]$ is the discount factor

Note 2: A more detailed definition of MDP involves specifying the initial state distribution $d_0(s_0)$ and augments either of the MDP definitions as:

Markov Decision Process with specified *initial state* is a 6-tuple $(\mathcal{S}, \mathcal{A}, T, r, d_0, \gamma)$, where:

- $\mathcal{S}$ is a set of states (finite or infinite, discrete, or continuous)
- $\mathcal{A}$ is a set of actions (finite or infinite, discrete, or continuous)
- $p(s'|s, a) \doteq \Pr\{S_t = s'|S_{t-1} = s, A_{t-1} = a\}$ is the probability to get from state s to state s' with action a.
- $r : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$ defines a *reward function*
- $d_0(s_0)$ defines the initial state distribution
- $\gamma \in [0,1]$ is the discount factor

Learning Policy (or just *Policy*): function $\pi : \mathcal{S} \to \mathcal{A}$ which represents mapping from states to probabilities of selecting each possible action.
If the agent is following policy π at time t, then $\pi(a|s)$ is the probability that $A_t = a$ if $S_t = s$. Note that $\pi(a|s)$ is an ordinary function which defines a probability distribution over $a \in \mathcal{A}(s)$ for each $s \in \mathcal{S}$.

We would like to modify the policy π with training or experience.

State-Value and Action-Value Functions

Let us assume that the current state is S_t, and actions are selected according to a stochastic policy π. Then we would like to derive an expression for the expectation of R_{t+1} in terms of π and $p(s', r|s, a)$.
Recall, the function $p(s', r|s, a)$ defines the dynamics of the MDP and is given as:

$$p(s', r|s, a) \doteq \Pr\{S_t = s', R_t = r|S_{t-1} = s, A_{t-1} = a\} \text{ for all } s', s \in \mathcal{S}, r \in \mathcal{R}, a \in \mathcal{A}(s) \qquad \text{(A.1)}$$

Then we can write:

$$\mathbb{E}_\pi[R_{t+1}|S_t = s] = \sum_a \pi(a|s) \sum_{s'} \sum_r p(s', r|s, a)\,[r] \qquad \text{(A.2)}$$

Here r denotes the reward of going from state s to state s' taking action a is given by MDP's R function: $r = R(s, s', a)$.
State-Value Function for Policy π (or simply *Value function;* aka *V function*): the value function of a state s under a policy π, denoted with $v_\pi(s)$, is the expected return when starting in s and following π thereafter. For MDPs, we can define v_π formally by

$$v_\pi(s) \doteq \mathbb{E}_\pi[G_t|S_t = s] = \mathbb{E}_\pi[\sum_{k=0}^{\infty} \gamma^k R_{t+k+1}|\, S_t = s] \text{ for all } s \in \mathcal{S} \qquad \text{(A.3)}$$

where $\mathbb{E}_\pi[\cdot]$ denotes the expected value of a random variable given that the agent follows policy π, and t is any time step. Note that the value of the terminal state, if any, is always zero.

Action-Value Function for Policy π (aka *Q function*):
We define the value of taking action a in state s under a policy π, denoted $q_\pi(s, a)$, as expected return starting from s, taking the action a, and thereafter following policy π:

$$q_\pi(s, a) \doteq \mathbb{E}_\pi[G_t|S_t = s, A_t = a] = \mathbb{E}_\pi[\sum_{k=0}^{\infty} \gamma^k R_{t+k+1}|\, S_t = s, A_t = a] \text{ for all } s \in \mathcal{S} \text{ and } a \in \mathcal{A}(s) \qquad \text{(A.4)}$$

Let us express v_π in terms of q_π and π. Given a state s, the state value function $v_\pi(s)$, given with (A.3), is equal to the expected cumulative return from that state given a distribution of actions π. The action value function q_π is the expectation of the return given state s, and taking action a as a starting point, and following policy π thereafter. Therefore, given a state s, the action-value function q_π is the weighted sum of the action-values over all relevant actions weighted by the policy weight:

$$v_\pi(s) = \sum_a \pi(a|s) q_\pi(s, a) \qquad \text{(A.5)}$$

Given a state s and an action a let us express the action-value function q_π in terms of the state value function v_π and the function defining the MDP dynamics $p(s', r|s, a)$. Recall, given a state s and an action a, the action value function q_π is given by the mathematical expectation of the discounted future rewards i.e. return G_t. The return G_t is the discounted sequence of rewards after the time step t and it can be written as:

$$G_t = \sum_{k=0}^{\infty} \gamma^k R_{t+k+1} = R_{t+1} + \gamma G_{t+1} \qquad \text{(A.6)}$$

It is important to recognize that

$$q_\pi(s, a) = \mathbb{E}_\pi[G_t|S_t = s, A_t = a] = \mathbb{E}_\pi[R_{t+1}|S_t = s, A_t = a] + \gamma \mathbb{E}_\pi[G_{t+1}|S_t = s, A_t = a]. \qquad \text{(A.7)}$$

The first term on the right-hand side of (A.7) can be expressed as:

$\mathbb{E}_\pi[R_{t+1}|S_t = s, A_t = a] = \sum_{s'} \sum_r p(s', r|s, a)\,[r].$ (A.8)

As before, r denotes the reward of going from state s to state s' taking action a is given by MDP's R function: $r = R(s, s', a)$.

The expectation in the second term on the right-hand side of (A.8) can be expressed as:

$\mathbb{E}_\pi[G_{t+1}|S_t = s, A_t = a] = \sum_{s'} \sum_r p(s', r|s, a) v_\pi(s').$ (A.9)

This is the expectation of the return starting at the next time step $t + 1$ following the policy π given the current state s and the action a, chosen according to π.
Substituting (A.8) and (A.9) into (A.7) gives us:

$q_\pi(s, a) = \sum_{s'} \sum_r p(s', r|s, a)\,[r + \gamma v_\pi(s')].$ (A.10)

Thus, the action-value function q_π given state s and action a following policy π is expressed as the sum of the next reward and discounted state-value weighted by probability distribution over the possible next states and next rewards from the given action a and state s.

Bellman's Equations for State-Value and Action-Value Functions

Bellman's equation for state values v

The value functions satisfy recursive relationships, and this property of the former will prove quite useful.
For any policy π and for any state s , the following consistency condition holds between the value of s and the value of its successor states. Starting with (A.6) applied to the definition of $v_\pi(s)$:

$v_\pi(s) \doteq \mathbb{E}_\pi[G_t|S_t = s] = \mathbb{E}_\pi[R_{t+1} + \gamma G_{t+1}|S_t = s]$ (A.11)

Using (5) the last equation becomes:

$$\begin{aligned} v_\pi(s) &= \sum_a \pi(a|s) \sum_{s'} \sum_r p(s', r|s, a)\left[r + \gamma \mathbb{E}_\pi[G_{t+1}|S_{t+1} = s']\right] \\ &= \sum_a \pi(a|s) \sum_{s'} \sum_r p(s', r|s, a)\,[r + \gamma v_\pi(s')] \quad \text{for all} \quad s \in \mathcal{S} \qquad \text{(A.12)} \end{aligned}$$

where it is implicit that the actions, a, are taken from the set $\mathcal{A}(s)$, that the next states, s', are taken from the set $\mathcal{S}$, and that the rewards, r, are taken from the set $\mathcal{R}$. Here the reward of going from state s to state s' taking action a is given by MDP's R function: $r = R(s, s', a)$. Note that the right-hand side of (A.12) is interpreted as an expected value obtained as a sum over the values of the triplet a, s', and r. For each triplet (a, s', r) the quantity $r + \gamma v_\pi(s')$ is weighed by its probability, $\pi(a|s)p(s', r|s, a)$.

Eq. (A.12) is known as the *Bellman equation* for v_π. It expresses a relationship between the value of a state and the values of its successor states.

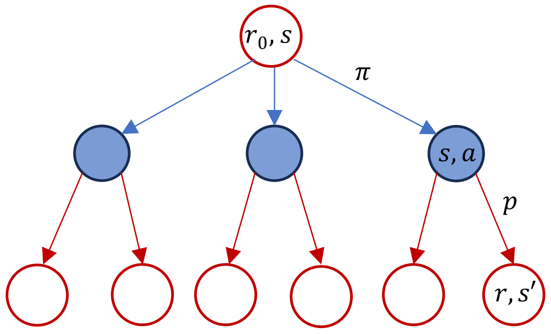

Figure A.1: Backup diagram for v_π

This relationship is expressed by the *Backup diagram* shown on Figure A.1. Each open circle, which will be denoted as *reward-state node* so forth, colored in red represents a state and the reward, which is associated with this state For instance, the root node shown on Figure 1 has associated reward r_0 and state s. Each solid circle, colored in blue represents a state-action pair and will be denoted as *state-action node* so forth. The specific state on the rightmost state-action node is shown as (s, a). Each directed blue edge connects state node with state-action node and represents application of the policy π to the root reward-state node (r_0, s). Each directed red edge emanating from a state-action node ends in a possible reward-state node corresponding to specific probable pair of reward r and new state s'. Thus, each directed red edge represents the application of the function p of the MDP dynamics. The Bellman equation (A.12) averages over all of the possibilities weighing each possibility represented by a path from the root of the Backup diagram on Figure A.1 to a leaf by its probability of occurring. It states that the value of the start state must equal the discounted value of the expected next state plus the reward expected along the way. The value function v_π is the unique solution to its Bellman equation. Various methods exist to compute exactly, approximate, or learn the value function.

Derivation of action value system of equations for discrete finite state

Let us assume that we are dealing with *discrete finite state* – that is we have $s_i \in \mathcal{S}\ \forall\ i \in [1, S]$. Here we have denoted $S = |\mathcal{S}|$.

Then from (A.3) we have:

$$v_\pi(s_i) \doteq \mathbb{E}_\pi[G_t|S_t = s_i] = \mathbb{E}_\pi[\textstyle\sum_{k=0}^{\infty} \gamma^k R_{t+k+1} |\, S_t = s_i] \text{ for all } s_i \in \mathcal{S} \qquad \text{(A.13)}$$

We also use (A.12) rewritten as:

$$v_\pi(s_i) = \textstyle\sum_a \pi(a|s_i) \sum_{j\neq i,r} p(s_j, r|s_i, a)\big[r + \gamma v_\pi(s_j)\big] + \sum_a \pi(a|s_i) \sum_r p(s_i, r|s_i, a)[r + \gamma v_\pi(s_i)]$$
$$\text{for all } s_i, s_j \in \mathcal{S} \qquad \text{(A.14)}$$

hence

$$v_\pi(s_i) - \gamma \textstyle\sum_a \pi(a|s_i) \sum_r p(s_i, r|s_i, a) \cdot v_\pi(s_i) - \gamma \sum_a \pi(a|s_i) \sum_{j\neq i,r} p(s_j, r|s_i, a) \cdot v_\pi(s_j)$$
$$= \textstyle\sum_a \pi(a|s_i) \sum_r p(s_i, r|s_i, a) \cdot r + \sum_a \pi(a|s_i) \sum_r p(s_i, r|s_i, a) \cdot r \qquad \text{(A.15)}$$

The left-hand side of (A2) can be rewritten as:

$$v_\pi(s_i) - \gamma \textstyle\sum_r \sum_a p(s_i, r|s_i, a) \cdot \pi(a|s_i)\, v_\pi(s_i) - \gamma \sum_{j\neq i,r} \sum_a p(s_j, r|s_i, a) \cdot \pi(a|s_i)\, v_\pi(s_j) \qquad \text{(A.16)}$$

The right-hand side of (A.15) are rearranged as :

$$\sum_r \sum_a p(s_i, r|s_i, a) \cdot \pi(a|s_i) \cdot r + \sum_{j\neq i, r} \sum_a p(s_j, r|s_i, a) \cdot \pi(a|s_i) \cdot r \qquad \text{(A.17)}$$

we denote with $p_\pi(s_j, r|s_i), p_\pi(s_j|s_i)$ and $r_\pi(s_j|s_i)$ the following expressions:

$$p_\pi(s_j, r|s_i) \doteq \mathbb{E}[\pi|S_{t-1} = s_i] = \sum_a p(s_j, r|s_i, a) \cdot \pi(a|s_i) \quad \text{(A.18)}$$

$$p_\pi(s_j|s_i) \doteq \sum_r p_\pi(s_j, r|s_i) \quad \text{(A.19)}$$

$$r_\pi(s_j|s_i) \doteq \sum_r p_\pi(s_j, r|s_i) \cdot r \quad \text{(A.20)}$$

For convenience we abbreviate:

$$p_{j,i} \doteq p_\pi(s_j|s_i)\,,\ r_i \doteq \sum_j r_\pi(s_j|s_i), S \doteq |\mathcal{S}| \qquad \text{(A.21)}$$

Using (A.16)-(A.21) in (3) leads to :

$$(1 - \gamma p_{i,i})v_\pi(s_i) - \gamma \sum_{j\neq i} p_{j,i} v_\pi(s_j) = r_i \qquad \text{(A.22)}$$

(A.22) represents a linear system of equations with respect to the $|\mathcal{S}|$ unknowns $v_\pi(s_i), s_i \in \mathcal{S}$.

Let us denote with $\boldsymbol{v_\pi}$ the column vector of the S unknowns $v_\pi(s_i), s_i \in \mathcal{S}$.
We denote with $\boldsymbol{A}$ the matrix formed by the elements and $\boldsymbol{b}$ the vector as shown below:

$$A = \begin{bmatrix} 1-\gamma p_{11} & p_{21} & \cdots & p_{S1} \\ p_{12} & 1-\gamma p_{22} & & p_{S2} \\ \vdots & & \ddots & \vdots \\ p_{1S} & p_{2S} & \cdots & 1-\gamma p_{SS} \end{bmatrix}, \quad b = \begin{bmatrix} r_1 \\ r_2 \\ \vdots \\ r_S \end{bmatrix} \qquad \text{(A.23)}$$

Then (A.22) in matrix form:

$$\boldsymbol{A} \cdot \boldsymbol{v_\pi} = \boldsymbol{b} \qquad \text{(A.24)}$$

Bellman's equation for action values $\boldsymbol{q}$

Let us derive a similar recursive relation with respect to the state-action value function. That is, we will find out what is the relation between the action value $q_\pi(s, a)$ and that for the possible successors to the state-action pair - $q_\pi(s', a')$. The derivation follows from the Backup diagram shown on Figure A.2 below.

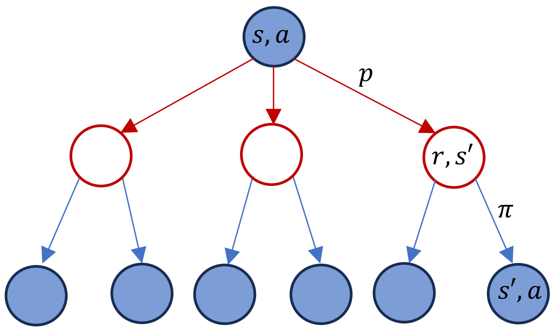

Figure A.2: Backup diagram for q_π

From (A.7) we can write:
$q_\pi(s,a) = \mathbb{E}_\pi[G_t | S_t = s, A_t = a] = \mathbb{E}_\pi[R_{t+1} + \gamma G_{t+1} | S_t = s, A_t = a]$ (A.25)

The expectation of the reward on the right-hand side can be rewritten as:

$\mathbb{E}_\pi[R_{t+1} + \gamma G_{t+1} | S_t = s, A_t = a] = \sum_{s',r} p(s', r | s, a)[r + \gamma q_\pi(s', a')]$ (A.26)

Here using (A.7) again we denote with $q_\pi(s', a')$ the expression for $\mathbb{E}_\pi[G_{t+1} | S_t = s, A_t = a]$.

Thus we get the Bellman's equation with respect q_π:

$q_\pi(s,a) = \sum_{s',r} p(s', r | s, a)[r + \gamma q_\pi(s', a')]$ (A.27)

Expressing the current state values v in terms of the next action values q

It is instructive to compare Eq (A.5) which we derived earlier with Eq (A.12) and Eq (A.27).
Eq (A.5) deserves its own Backup diagram shown on Figure 3:

$v_\pi(s) = \sum_a \pi(a|s) q_\pi(s,a)$ (A.5)

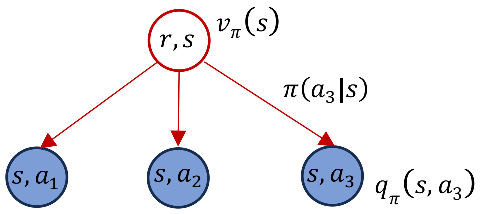

Figure A.3: Backup diagram for relation between $v_\pi(s)$ and $q_\pi(s,a)$

Eq (A.5) tells us how the value of a state depends on the values of the actions possible in that state and on how likely each action is to be taken under the current policy. The state value which corresponds to the state-value node at the root is obviously $v_\pi(s)$ and the action values which corresponds to its children are $q_\pi(s, a_i), i = 1..3$. The probability with which each action a_i is taken is given by the policy i.e. $\pi(a_i|s), i = 1..3$.

Expressing the current action values q in terms of the next state values v

The value of an action, $q_\pi(s, a)$, depends on the expected next reward and the expected sum of the remaining rewards. Expressed as a Backup diagram we arrive at Figure A.4 shown below.

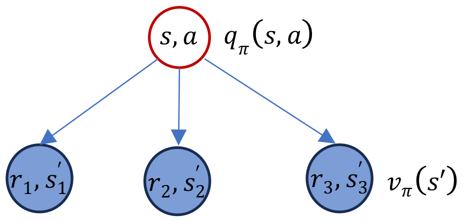

Figure A.4: Backup diagram expressing the dependence of the current action value on the expected next reward-state values.

Formally expressed this relation becomes:

From Eq (A.7) we have
$q_\pi(s, a) = \mathbb{E}_\pi[G_t | S_t = s, A_t = a] = \mathbb{E}_\pi[R_{t+1} | S_t = s, A_t = a] + \gamma \mathbb{E}_\pi[G_{t+1} | S_t = s, A_t = a]$

Clearly, $\mathbb{E}_\pi[R_{t+1} | S_t = s, A_t = a] = \sum_{s',r} p(s', r | s, a) \cdot r$ where $r = R(s, s', a)$
Eq. (A.9) states that the expectation in the second term of (A.7) can be written as:

$\mathbb{E}_\pi[G_{t+1} | S_t = s, A_t = a] = \sum_{s',r} p(s', r | s, a) \cdot v_\pi(s')$

Combining the last two results we obtain:

$q_\pi(s, a) = \mathbb{E}[R_{t+1} + \gamma v_\pi(s') | S_t = s, A_t = a] = \sum_{s',r} p(s', r | s, a)[r + \gamma v_\pi(s')]$ (A.28)

Bibliography

Reinforcement Learning, Richard S. Sutton, Andrew G. Barto, second edition, 2020