ECE 1508: Reinforcement Learning

Chapter 2: Model-based RL

Ali Bereyhi

ali.bereyhi@utoronto.ca

Department of Electrical and Computer Engineering

University of Toronto

Summer 2024

Classical RL Methods: *Recall*

Ultimate goal in an RL problem is to find the optimal policy

As mentioned, we have *two major challenges* in this way

1. *We need to compute values explicitly*
2. *We often deal with settings with huge state spaces?*

In this part of the course, we are going to handle the first challenge

- *This chapter* $\rightsquigarrow$ *Model-based methods*
- *Next chapter* $\rightsquigarrow$ *Model-free methods*

A Good Start Point: *Model-based RL*

In a nutshell, in model-based methods

> *we are able to describe mathematically the behavior of environment*

This might come from the *nature of problem* or *simply postulated by us*

Model-based RL

- Bellman Equation
  - value iteration
  - policy iteration

Model-free RL

- on-policy methods
  - temporal difference
  - Monte Carlo
  - SARSA
- off-policy methods
  - Q-learning

Complete State is *Markov Process*

When we formulated the RL framework, we stated that

a complete state must describe a Markov process

Markov Process

Sequence $S_1 \to S_2 \to \dots$ describe a Markov process if

$$\Pr\{S_{t+1} = s_{t+1} | S_t = s_t, \dots, S_1 = s_1\} = \Pr\{S_{t+1} = s_{t+1} | S_t = s_t\}$$

Following this fact, we introduced the concepts of

rewarding and transition functions

Recall: *Transition and Rewarding*

Both these mappings only depend on current state and action

Transition function maps state S_t and action A_t to the next state S_{t+1}

$$\mathcal{P}(\cdot) : \mathbb{S} \times \mathbb{A} \mapsto \mathbb{S}$$

Rewarding function maps state S_t and action A_t to reward R_{t+1}

$$\mathcal{R}(\cdot) : \mathbb{S} \times \mathbb{A} \mapsto \left\{r^1, \ldots, r^L\right\}$$

We said that *these mappings are in general random*

Describing Markov Trajectory

Markovity of the state indicates that we observe the following trajectory

$$S_0, A_0 \to (R_1, S_1), A_1 \to \dots \to (R_t, S_t), A_t \to (R_{t+1}, S_{t+1})$$

This trajectory describes a Markov process with conditional distribution

$$\begin{aligned} p(r, \bar{s}|s, a) &= \Pr\{R_{t+1} = r, S_{t+1} = \bar{s}|S_t = s, A_t = a\} \\ &= \Pr\{R_t = r, S_t = \bar{s}|S_{t-1} = s, A_{t-1} = a\} \\ &\vdots \\ &= \Pr\{R_1 = r, S_1 = \bar{s}|S_0 = s, A_0 = a\} \end{aligned}$$

$\boxed{\textit{The above trajectory describes a Markov Decision Process (MDP)}}$

Finite MDPs

In this course, we focus on *finite* MDPs

Finite MDP

The Markov process

$$S_0, A_0 \to (R_1, S_1), A_1 \to \dots \to (R_t, S_t), A_t$$

is a finite MDP if rewards, actions and states belong to a finite set, i.e.,

$$r \in \{r^1, \dots, r^L\} \qquad a \in \{a^1, \dots, a^M\} \qquad s \in \{s^1, \dots, s^N\}$$

MDPs are completely described by conditional distribution $p(r, \bar{s}|s, a)$

We call $p(r, \bar{s}|s, a)$ *hereafter rewarding-transition model*

Model-based RL via *MDP*

+ *What makes it now model-based RL?*
– We assume that *rewarding-transition model* $p(r, \bar{s}|s, a)$ is given to us
+ *But you said for model-based RL, we should know the transition and rewarding functions!*
– Well, we can describe them using $p(r, \bar{s}|s, a)$!

Rewarding Model

Assume we are in state $S_t = s$ and act $A_t = a$; then, R_{t+1} is a random variable whose distribution is given by

$$p(r|s, a) = \sum_{n=1}^{N} p(r, s^n|s, a)$$

We call this distribution hereafter rewarding model

Model-based RL via *MDP*

Similarly, we can describe the *transition function*

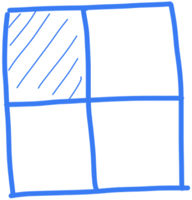
Transition Model

Assume we are in state $S_t = s$ and act $A_t = a$; then, next state S_{t+1} is a random variable whose distribution is given by

$$p\left(\bar{s}|s, a\right) = \sum_{\ell=1}^{L} p\left(r^{\ell}, \bar{s}|s, a\right)$$

We call this distribution hereafter transition model

Example: *Dummy Grid World*



We have a grid board where at each cell we can move

$$\mathbb{A} = \{0 \equiv \texttt{left}, 1 \equiv \texttt{down}, 2 \equiv \texttt{right}, 3 \equiv \texttt{up}\}$$

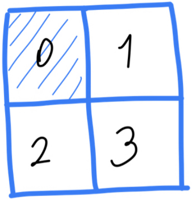
Our ultimate goal is to arrive at top-left corner

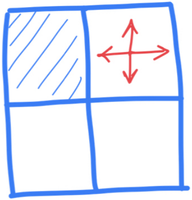
through shortest path

This problem describes an MDP with deterministic rewarding-transition model

- *State is the number of cell*
- *Action is the direction we move*
- *Reward is -1 each time we move until we get to destination*
  - ↳ *Reward is -0.5 when we hit the corners*

Example: *Dummy Grid World*





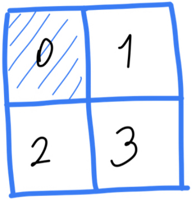
$\mathbb{S} = \{0, 1, 2, 3\}$

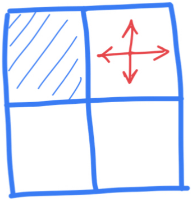
$\mathbb{A} = \{0 \equiv \texttt{left}, 1 \equiv \texttt{down}, 2 \equiv \texttt{right}, 3 \equiv \texttt{up}\}$

Let's write the rewarding-transition model down

$$p(r, \bar{s} | 3, 3) = \begin{cases} 1 & (r, \bar{s}) = (-1, 1) \\ 0 & (r, \bar{s}) \neq (-1, 1) \end{cases}$$

Example: *Dummy Grid World*





$\mathbb{S} = \{0, 1, 2, 3\}$

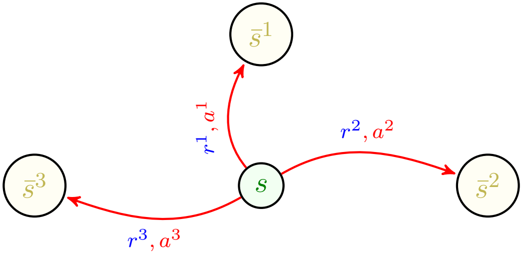
$\mathbb{A} = \{0 \equiv \texttt{left}, 1 \equiv \texttt{down}, 2 \equiv \texttt{right}, 3 \equiv \texttt{up}\}$

Let's write the rewarding-transition model down

$$p(r, \bar{s}|0, a) = \begin{cases} 1 & (r, \bar{s}) = (0, 0) \\ 0 & (r, \bar{s}) \neq (0, 0) \end{cases} \rightsquigarrow s = 0 \text{ is terminal state}$$

Transition Diagram

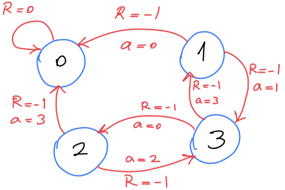
It is sometimes helpful to show transition model via a *transition diagram*



This diagram describes a graph

- *Each node is a possible state: we have in total N nodes*
- *Node s is connected to $\bar{s}$ if the probability of transition is non-zero*
  - ↳ *We could specify the action that can lead us to the new state*
  - ↳ *The graph could have loops including self-loop*

Transition Diagram: *Dummy Grid World*



In our dummy grid world, we have four states

- *If we are in terminal state we always remain there with no rewards*
- *From state $s = 1$ we can go to states $\bar{s} = 0, 3$ depending on action*
  ↳ *We can also remain in state $s = 1$ and reward with -0.5 if we hit corners*
- *From state $s = 2$ we can go to states $\bar{s} = 0, 3$ depending on action*
  ↳ *We can also remain in state $s = 1$ and reward with -0.5 if we hit corners*
- *From state $s = 3$ we can go to states $\bar{s} = 1, 2$ depending on action*
  ↳ *We can also remain in state $s = 1$ and reward with -0.5 if we hit corners*

Expected Action Reward

As we said, *using rewarding-transition model we can describe the environment completely: for instance, let's see what would be the expected immediate reward that we get if in state s we act a*

$$\begin{aligned}
\bar{\mathcal{R}}(s,a) &= \mathbb{E}\left\{R_{t+1} | s,a\right\} \rightsquigarrow \text{we simplify notation } S_t = s \text{ to } s \\
&= \sum_{\ell=1}^{L} r^{\ell} p\left(r^{\ell} | s,a\right) \\
&= \sum_{\ell=1}^{L} r^{\ell} \sum_{n=1}^{N} p\left(r^{\ell}, s^{n} | s,a\right) \\
&= \sum_{\ell=1}^{L} \sum_{n=1}^{N} r^{\ell} p\left(r^{\ell}, s^{n} | s,a\right) \rightsquigarrow \text{rewarding-transition model}
\end{aligned}$$

Expected Action Reward

$\bar{\mathcal{R}}(s, a)$ describes

the reward we expect to see immediately after acting a in state s

We are going to see this expectation a lot, so maybe we could give it a name

Expected Action Reward

The expected reward for a state-action pair (s, a) is defined as

$$\bar{\mathcal{R}}(s, a) = \mathbb{E}\left\{R_{t+1} | s, a\right\} = \sum_{\ell=1}^{L} \sum_{n=1}^{N} r^{\ell} p\left(r^{\ell}, s^{n} | s, a\right)$$

Obviously, $\bar{\mathcal{R}}(s, a)$ does not depend on policy

Expected Policy Reward

+ *Can we relate it also to our policy?*
- Sure! We could average over our policy

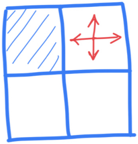
Expected Policy Reward

The expected immediate reward of policy π at state s is defined as

$$\bar{\mathcal{R}}_\pi(s) = \mathbb{E}_\pi\left\{R_{t+1}|s\right\} = \sum_{m=1}^{M} \mathbb{E}\left\{R_{t+1}|s, a^m\right\} \pi\left(a^m|s\right)$$
$$= \sum_{m=1}^{M}\sum_{\ell=1}^{L}\sum_{n=1}^{N} r^\ell p\left(r^\ell, s^n|s, a\right) \pi\left(a^m|s\right)$$

It describes reward we expect to see immediately after state s while playing π

Example: *Dummy Grid World*



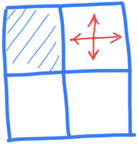
In our dummy grid world, we can easily compute the expected immediate reward

$$\bar{\mathcal{R}}(1, a) = \begin{cases} -1 & a \in \{0, 1\} \\ -0.5 & a \in \{2, 3\} \end{cases}$$

Obviously in terminal state we always get zero expected reward, e.g., for all a

$$\bar{\mathcal{R}}(0, a) = 0$$

Example: *Dummy Grid World*



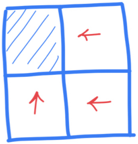
Now assume that we play uniformly at random, i.e., for all a and s

$$\pi\left(a|s\right) = \frac{1}{4}$$

In this case the expected policy reward is

$$\bar{\mathcal{R}}_\pi\left(1\right) = \sum_{a=0}^{3} \bar{\mathcal{R}}\left(1, a\right)\pi\left(a|1\right) = -0.75$$

Example: *Dummy Grid World*



But if we change to above deterministic policy: the expected reward changes to

$$\bar{\mathcal{R}}_\pi\left(1\right) = \sum_{a=0}^{3} \bar{\mathcal{R}}\left(1, a\right) \pi\left(a|1\right) = \bar{\mathcal{R}}\left(1, 0\right) = -1$$

and we can easily show that

$$\bar{\mathcal{R}}_\pi\left(0\right) = 0 \qquad \bar{\mathcal{R}}_\pi\left(2\right) = -1 \qquad \bar{\mathcal{R}}_\pi\left(3\right) = -1$$

Computing Value Functions: *Naive Approach*

Now that we have a concrete model for our environment: *we should go ahead and compute the value function, as we want to optimize it*

Let's start with direct computation

$$\begin{aligned} v_\pi(s) &= \mathbb{E}_\pi\left\{G_t | s\right\} \\ &= \mathbb{E}_\pi\left\{R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+3} + \ldots | s\right\} \\ &= \mathbb{E}_\pi\left\{R_{t+1} | s\right\} + \gamma \mathbb{E}_\pi\left\{R_{t+2} | s\right\} + \gamma^2 \mathbb{E}_\pi\left\{R_{t+3} | s\right\} + \ldots \\ &= \bar{\mathcal{R}}_\pi(s) + \gamma \mathbb{E}_\pi\left\{R_{t+2} | s\right\} + \gamma^2 \mathbb{E}_\pi\left\{R_{t+3} | s\right\} + \ldots \end{aligned}$$

+ *How can we compute next terms?*
– *We could use the rewarding-transition model of MDP*

Computing Value Functions: *Naive Approach*

Let's try the second term for example: we first define the notation

$$\mathbb{E}_\pi \left\{ R_{t+2} | s, s^n, a^m, a^j \right\} = \mathbb{E}_\pi \left\{ R_{t+2} | S_t = s, S_{t+1} = s^n, A_t = a^m, A_{t+1} = a^j \right\}$$

We can easily compute $\mathbb{E}_\pi \left\{ R_{t+2} | s, s^n, a^m, a^j \right\}$ *as*

$$\begin{aligned} \mathbb{E}_\pi \left\{ R_{t+2} | s, s^n, a^m, a^j \right\} &= \sum_{\ell=1}^{L} r^\ell p \left(r^\ell | s, s^n, a^m, a^j \right) \\ &= \sum_{\ell=1}^{L} r^\ell p \left(r^\ell | s^n, a^j \right) \end{aligned}$$

Computing Value Functions: *Naive Approach*

We can then say that

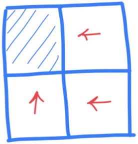
$$\mathbb{E}_\pi \left\{ R_{t+2} | s \right\} = \sum_{n=1}^{N} \sum_{m=1}^{M} \sum_{j=1}^{M} \mathbb{E}_\pi \left\{ R_{t+2} | s, s^n, a^m, a^j \right\} p\left(a^m, s^n, a^j | s \right)$$

and write down $p\left(a^m, s^n, a^j | s \right)$ *using chain rule*

$$\begin{aligned} p\left(a^m, s^n, a^j | s \right) &= p\left(a^m | s \right) p\left(s^n | s, a^m \right) p\left(a^j | s, a^m, s^n \right) \\ &= \pi\left(a^m | s \right) \underbrace{p\left(s^n | s, a^m \right)}_{\text{transition model}} \pi\left(a^j | s^n \right) \end{aligned}$$

+ *How can we compute the next term?*
– We should repeat the same approach: *there will be more nested sums*

Example: *Dummy Grid World*



Let's start with the above policy: π^1

$$v_{\pi^1}(1) = \mathbb{E}_{\pi^1}\left\{R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+3} + \ldots | 1\right\}$$

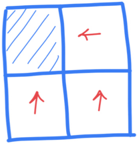
Following policy at $s = 1$ *we end up at terminal state at next time*

$$v_{\pi^1}(1) = \mathbb{E}_{\pi^1}\left\{R_{t+1} + \gamma 0 + \gamma^2 0 + \ldots | 1\right\} = \bar{\mathcal{R}}_{\pi^1}(1) = -1$$

Same way, we can conclude that

$$v_{\pi^1}(0) = 0 \qquad v_{\pi^1}(2) = -1 \qquad v_{\pi^1}(3) = -2$$

Example: *Dummy Grid World*

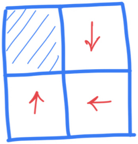


Let's change to policy π^2: we could follow same steps to show that

$$v_{\pi^2}(0) = 0 \qquad v_{\pi^2}(1) = -1 \qquad v_{\pi^2}(2) = -1 \qquad v_{\pi^2}(3) = -2$$

We note that it returns the same values as policy π^1

Example: *Dummy Grid World*



Let's now look at policy π^3: we could follow same steps to show that

$$v_{\pi^3}(0) = 0 \qquad v_{\pi^3}(1) = -3 \qquad v_{\pi^3}(2) = -1 \qquad v_{\pi^3}(3) = -2$$

We can see that

$$\pi^1 = \pi^2 \geqslant \pi^3$$

Computing Value Functions: *Practical Approach*

+ *But, we should compute infinite terms in general!*
– Well, if we are lucky: *the sequence either terminates or shows a pattern*
+ *What if that doesn't happen?*
– Then, this approach really does not work!

This is why we called it the naive approach, since we never use this approach: in practice,

we always invoke a Bellman equation

and find it via dynamic programming

Future Return: *Recursive Property*

Even though future return looks infinte, it has a simple recursive property

$$
\begin{aligned}
G_t &= \sum_{i=0}^{\infty} \gamma^i R_{t+i+1} \\
&= R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+3} + \dots \\
&= R_{t+1} + \gamma \left(R_{t+2} + \gamma R_{t+3} + \dots \right) \\
&= R_{t+1} + \gamma G_{t+1}
\end{aligned}
$$

We can use this property to find a fixed-point equation for the value function!

Value Function: *Recursive Property*

Say we are playing with policy π: *we can write the value function as*

$$\begin{aligned} v_\pi(s) &= \mathbb{E}_\pi \left\{ G_t | s \right\} \\ &= \mathbb{E}_\pi \left\{ R_{t+1} + \gamma G_{t+1} | s \right\} \\ &= \mathbb{E}_\pi \left\{ R_{t+1} | s \right\} + \gamma \mathbb{E}_\pi \left\{ G_{t+1} | s \right\} \\ &= \bar{\mathcal{R}}_\pi(s) + \gamma \underbrace{\mathbb{E}_\pi \left\{ G_{t+1} | s \right\}}_{?} \end{aligned}$$

+ *Isn't that term again the value function at s?*
– Be careful! It's not

Attention

The second term is not the value of state s

$$\mathbb{E}_\pi \left\{ G_{t+1} | s \right\} = \mathbb{E}_\pi \left\{ G_{t+1} | S_t = s \right\} \neq \mathbb{E}_\pi \left\{ G_t | S_t = s \right\} = v_\pi(s)$$

Value Function: *Recursive Property*

Let's do some marginalization

$$
\begin{aligned}
\mathbb{E}_\pi \left\{ G_{t+1} | s \right\} &= \sum_{n=1}^{N} \mathbb{E}_\pi \left\{ G_{t+1} | S_t = s, S_{t+1} = s^n \right\} \Pr \left\{ S_{t+1} = s^n | S_t = s \right\} \\
&= \sum_{n=1}^{N} \mathbb{E}_\pi \left\{ G_{t+1} | s, s^n \right\} p \left(s^n | s \right)
\end{aligned}
$$

Well, we need to specify the two terms in under summation, i.e.,

- $\mathbb{E}_\pi \left\{ G_{t+1} | s, s^n \right\}$
- $p \left(s^n | s \right) = \Pr \left\{ S_{t+1} = s^n | S_t = s \right\}$

Value Function: *Recursive Property*

Recall the trajectory

$$S_0, A_0 \rightarrow (R_1, S_1), A_1 \rightarrow \dots \rightarrow (R_{t+1}, S_{t+1}), A_{t+1} \rightarrow (R_{t+2}, S_{t+2})$$

If we know state S_{t+1} any reward after $t+1$ only depends on S_{t+1}, i.e.,

$$\mathbb{E}_\pi \left\{ G_{t+1} \middle| S_t = s, S_{t+1} = s^n \right\} = \mathbb{E}_\pi \left\{ G_{t+1} \middle| S_{t+1} = s^n \right\}$$

This indicates that

$$\mathbb{E}_\pi \left\{ G_{t+1} \middle| s, s^n \right\} = v_\pi \left(s^n \right)$$

I.e., the value function at state s^n

Value Function: *Recursive Property*

We can further find $p\left(s^n|s\right)$ from transition model and policy

$$\begin{aligned} p_\pi\left(s^n|s\right) &= \sum_{m=1}^{M} p\left(s^n, a^m|s\right) \\ &= \sum_{m=1}^{M} p\left(a^m|s\right) p\left(s^n|a^m, s\right) \\ &= \sum_{m=1}^{M} \pi\left(a^m|s\right) p\left(s^n|s, a^m\right) \rightsquigarrow \text{depends on policy} \end{aligned}$$

We know have both terms in terms of transition model and policy

Value Function: *Recursive Property*

Replacing into the equation, where we left we have

$$
\begin{aligned}
\mathbb{E}_\pi \left\{ G_{t+1} | s \right\} &= \sum_{n=1}^{N} \mathbb{E}_\pi \left\{ G_{t+1} | s, s^n \right\} p \left(s^n | s \right) \\
&= \sum_{n=1}^{N} v_\pi \left(s^n \right) p_\pi \left(s^n | s \right) \\
&= \sum_{n=1}^{N} \sum_{m=1}^{M} v_\pi \left(s^n \right) p \left(s^n | s, a^m \right) \pi \left(a^m | s \right)
\end{aligned}
$$

We can also present it by shorter notation as

$$
\mathbb{E}_\pi \left\{ G_{t+1} | s \right\} = \mathbb{E}_\pi \left\{ v_\pi \left(S_{t+1} \right) | s \right\}
$$

Value Function: *Recursive Property*

Back to computation of value function, we have

$$\begin{aligned} v_\pi\left(s\right) &= \bar{\mathcal{R}}_\pi\left(s\right) + \gamma \mathbb{E}_\pi\left\{G_{t+1} | s\right\} \\ &= \bar{\mathcal{R}}_\pi\left(s\right) + \gamma \mathbb{E}_\pi\left\{v_\pi\left(S_{t+1}\right) | s\right\} \\ &= \bar{\mathcal{R}}_\pi\left(s\right) + \gamma \sum_{n=1}^{N} v_\pi\left(s^n\right) p_\pi\left(s^n | s\right) \end{aligned}$$

This is a recursive equation that relates value of one state to other values

which is a Bellman equation

Bellman Equation: *Value*

Bellman Equation for Value Function

For any policy π the value function at each state s satisfies

$$v_\pi(s) = \bar{\mathcal{R}}_\pi(s) + \gamma \sum_{n=1}^{N} v_\pi(s^n)\, p_\pi(s^n | s)$$

+ *Well! What is the use of Bellman equation?*
– It describes a *fixed-point equation that can be solved for $v_\pi(s)$!*

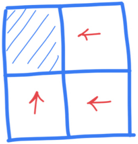
Bellman Equation: *Breaking Down*

$$\boxed{v_\pi\left(s\right) = \bar{\mathcal{R}}_\pi\left(s\right) + \gamma \sum_{n=1}^{N} v_\pi\left(s^n\right) p_\pi\left(s^n | s\right)}$$

In general, we have N possible state $\rightsquigarrow$ *we have N possible values*

- *Bellman equation relates each value to other $N-1$ values*
  - ↳ *For each s*, Bellman equation has *N unknowns* $v_\pi\left(s^1\right), \ldots, v_\pi\left(s^N\right)$
- *We can write the Bellman equation for all N states*
  - ↳ *We have N equations each with N unknowns*
- *We solve this system of equations for unknowns $v_\pi\left(s^1\right), \ldots, v_\pi\left(s^N\right)$*

Example: *Dummy Grid World*



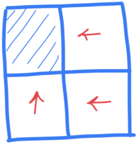
Let's try with our dummy grid world: we saw that

$$\bar{\mathcal{R}}_\pi(0) = 0 \qquad \bar{\mathcal{R}}_\pi(1) = -1 \qquad \bar{\mathcal{R}}_\pi(2) = -1 \qquad \bar{\mathcal{R}}_\pi(3) = -1$$

Now let's consider the values unknown

$$v_\pi(0), v_\pi(1), v_\pi(2), v_\pi(3)$$

Example: *Dummy Grid World*



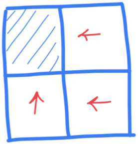
We set $\gamma = 1$ *and start with state* $s = 0$

$$v_\pi(0) = \bar{\mathcal{R}}_\pi(0) + \sum_{\bar{s}=0}^{3} v_\pi(\bar{s})\, p_\pi(\bar{s}|0)$$

We know that

$$p_\pi(\bar{s}|0) = \begin{cases} 1 & \bar{s} = 0 \\ 0 & \bar{s} \neq 0 \end{cases}$$

Example: *Dummy Grid World*

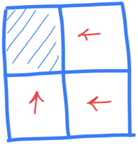


This concludes that at state $s = 0$, *Bellman equation reads*

$$v_\pi(0) = 0 + v_\pi(0)$$

which is an obvious equation; let's try $s = 1$

Example: *Dummy Grid World*



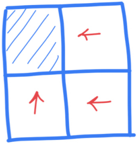
At state $s = 0$, we have

$$v_\pi(1) = \bar{\mathcal{R}}_\pi(1) + \sum_{\bar{s}=0}^{3} v_\pi(\bar{s}) \, p_\pi(\bar{s}|1)$$

Again we can easily say based on the policy that

$$p_\pi(\bar{s}|1) = \begin{cases} 1 & \bar{s} = 0 \\ 0 & \bar{s} \neq 0 \end{cases}$$

Example: *Dummy Grid World*



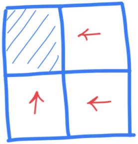
This concludes that at state $s = 1$, *Bellman equation reads*

$$v_\pi(1) = -1 + v_\pi(0)$$

which relates $v_\pi(1)$ *to* $v_\pi(0)$. *If we keep repeating we get further*

$$v_\pi(2) = -1 + v_\pi(0)$$
$$v_\pi(3) = -1 + v_\pi(2)$$

Example: *Dummy Grid World*



We now have the system of equations

$$
\begin{aligned}
v_\pi(1) &= -1 + v_\pi(0) \\
v_\pi(2) &= -1 + v_\pi(0) \\
v_\pi(3) &= -1 + v_\pi(2)
\end{aligned}
$$

We also know that $s = 0$ is a terminal state, and thus $v_\pi(0) = 0$: so, we get

$$
v_\pi(1) = -1 \qquad v_\pi(2) = -1 \qquad v_\pi(3) = -2
$$

Bellman Equation: *Action-Value*

We can find a Bellman equation for Action-value function as well: *say we play with policy* π

$$
\begin{aligned}
q_\pi(s,a) &= \mathbb{E}_\pi\left\{G_t \middle| s,a\right\} \\
&= \mathbb{E}_\pi\left\{R_{t+1} + \gamma G_{t+1} \middle| s,a\right\} \\
&= \mathbb{E}\left\{R_{t+1} \middle| s,a\right\} + \gamma \mathbb{E}_\pi\left\{G_{t+1} \middle| s,a\right\} \\
&= \bar{\mathcal{R}}(s,a) + \gamma \underbrace{\mathbb{E}_\pi\left\{G_{t+1} \middle| s,a\right\}}_{?}
\end{aligned}
$$

We need to compute

$$
\mathbb{E}_\pi\left\{G_{t+1} \middle| s,a\right\}
$$

in terms of the rewarding-transition model and policy

Action-Value: *Recursive Property*

We apply the marginalization trick

$$\mathbb{E}_\pi \left\{ G_{t+1} | s, a \right\} = \sum_{n=1}^{N} \mathbb{E}_\pi \left\{ G_{t+1} | S_t = s, S_{t+1} = s^n, A_t = a \right\} p\left(s^n | s, a \right)$$

Attention

Recalling the trajectory of the MDP, we should note that

$$q_\pi \left(s^n, a \right) \neq \mathbb{E}_\pi \left\{ G_{t+1} | S_t = s, S_{t+1} = s^n, A_t = a \right\} = v_\pi \left(s^n \right)$$

In fact, once we know S_{t+1}, the previous action does not contain any extra information! We only gain information, if we observe A_{t+1}, i.e.,

$$\mathbb{E}_\pi \left\{ G_{t+1} | S_t = s, S_{t+1} = s^n, A_{t+1} = a \right\} = q_\pi \left(s^n, a \right)$$

Action-Value: *Recursive Property*

So, we can replace it into original equation to get

$$\begin{aligned}\mathbb{E}_\pi \left\{ G_{t+1} | s, a \right\} &= \sum_{n=1}^{N} \mathbb{E}_\pi \left\{ G_{t+1} | S_t = s, S_{t+1} = s^n, A_t = a \right\} p\left(s^n | s, a \right) \\ &= \sum_{n=1}^{N} v_\pi \left(s^n \right) p \left(s^n | s, a \right)\end{aligned}$$

This implies that

$$\begin{aligned}q_\pi \left(s, a \right) &= \bar{\mathcal{R}} \left(s, a \right) + \gamma \sum_{n=1}^{N} v_\pi \left(s^n \right) p \left(s^n | s, a \right) \\ &= \bar{\mathcal{R}} \left(s, a \right) + \gamma \mathbb{E} \left\{ v_\pi \left(S_{t+1} \right) | s, a \right\}\end{aligned}$$

Bellman Equation: *Action-Value*

Bellman Equation I for Action-Value Function

For any policy π the action-value function at each pair (s, a) satisfies

$$q_\pi (s, a) = \bar{\mathcal{R}} (s, a) + \gamma \sum_{n=1}^{N} v_\pi (s^n) \, p (s^n | s, a)$$

After doing Assignment 1, you will immediately conclude the following extension

Bellman Equation II for Action-Value Function

For any policy π the action-value function at each pair (s, a) satisfies

$$q_\pi (s, a) = \bar{\mathcal{R}} (s, a) + \gamma \sum_{n=1}^{N} \sum_{m=1}^{M} q_\pi (s^n, a^m) \, \pi (a^m | s^n) \, p (s^n | s, a)$$

Computing Action-Value via Bellman Equation

We can again use the recursive equation

$$q_\pi(s,a) = \bar{\mathcal{R}}(s,a) + \gamma \sum_{n=1}^{N} \sum_{m=1}^{M} q_\pi(s^n, a^m)\, \pi(a^m|s^n)\, p(s^n|s,a)$$

to find the action-value function: *we have in this case NM possible values*

- *Bellman equation relates each action-value to other action-values*
  - ↳ *For each s and a, Bellman equation has NM unknowns $q_\pi(s^n, a^m)$*
- *We can write the Bellman equation for all NM cases*
- *We solve this system of equations for unknowns $q_\pi(s^n, a^m)$*

Bellman Equation: *Backup Diagram*

Bellman equation gives an

interesting visualization for values and action-values

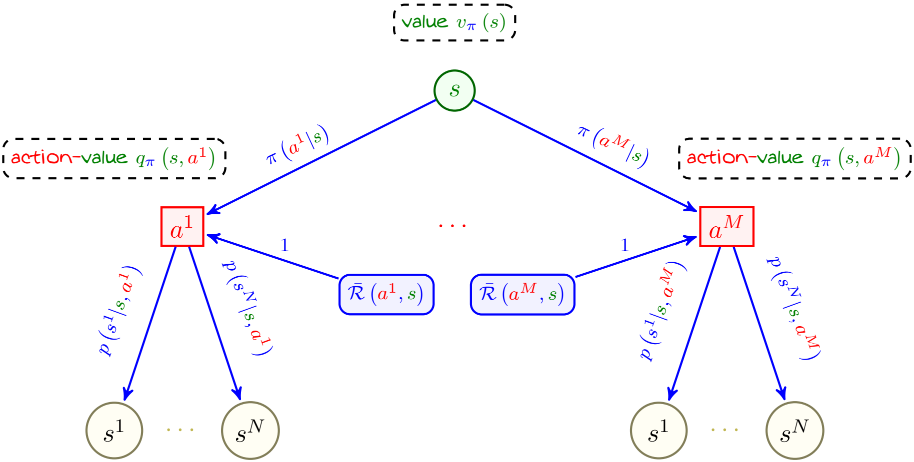
which can be shown in the s-called *backup diagram*

For simplicity, we consider $\gamma = 1$ in the backup diagram

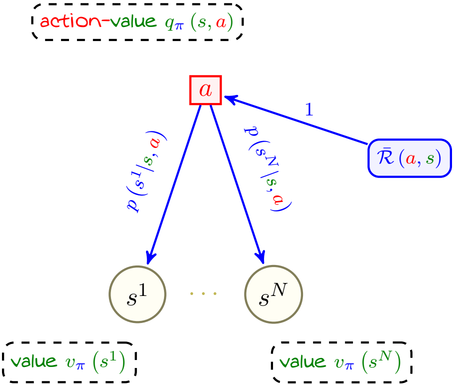
- *Each circle node is a state and carries the value of the state*
- *Each square node is an action and carries the action-value of the pair*
- *Each edge is a transition and carries a probability*
- *As we pass from leaves to root*
  - *Value of each node multiplies to its probability on the edge*
  - *They add up when they meet at a parent node*

    ↳ *This makes the value of the parent node*

Backup Diagram: *For Given Policy*



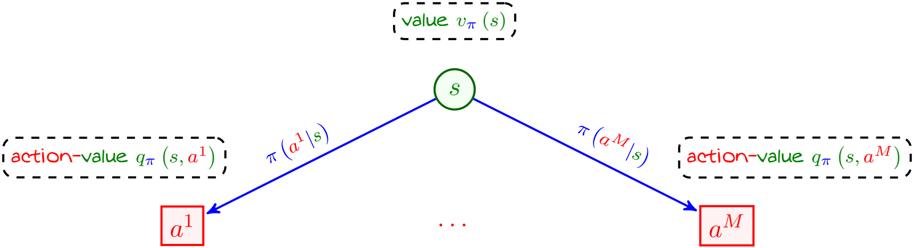
Backup Diagram: *For Given Policy*



Let's look at it part by part: *first we pass from leaves to action parent*

$$q_\pi(s, a) = \bar{\mathcal{R}}(s, a) + \sum_{n=1}^{N} v_\pi(s^n)\, p(s^n|s, a)$$

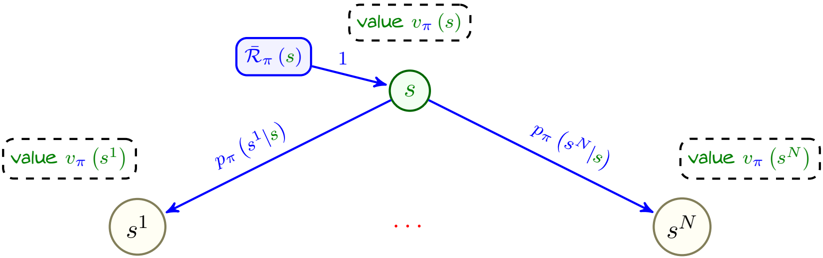
Backup Diagram: *For Given Policy*



Then, we pass from action parents to the root state

$$v_\pi(s) = \sum_{m=1}^{M} \pi(a^m|s)\, q_\pi(s, a^m)$$

Backup Diagram: *For Given Policy*



We could also have its alternative form expected over actions

$$v_\pi(s) = \bar{\mathcal{R}}_\pi(s) + \sum_{n=1}^{N} p_\pi(s^n|s)\, v_\pi(s^n)$$

Finding Optimal Values

+ *Well! Bellman lets us compute value of a given policy. But, how can we find the optimal value? It doesn't seem to solve this problem!*
- We can in fact use it to directly find the optimal values!
+ *That sounds a bit weird!*
- Once we know the *optimality constraint*, it doesn't anymore

Optimal Value: *Optimality Constraint*

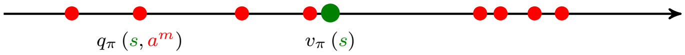
In Assignment 1, you show that *for any state we have*

$$v_\pi(s) = \sum_{m=1}^{M} q_\pi(s, a^m)\, \pi(a^m | s)$$

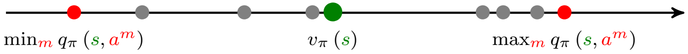
Now, recall that *policy is a conditional distribution meaning that*

$$0 \leqslant \pi(a^m | s) \leqslant 1$$

We can think of it as



Optimal Value: *Optimality Constraint*



It is hence obvious that

$$\min_m q_\pi(s, a^m) \leqslant v_\pi(s) \leqslant \max_m q_\pi(s, a^m)$$

We can use this simple fact to find a constraint on optimal values

Optimal Value: *Optimality Constraint*

If our policy is the optimal policy; then, we should have

$$v_\star(s) = \text{maximum possible value} = \max_m q_\star(s, a^m)$$

+ *But, can we guarantee that we can achieve such value?*
– Sure! We can set an optimal policy to

$$\pi^\star(a^m|s) = \begin{cases} 1 & m = \operatorname*{argmax}_m q_\star(s, a^m) \\ 0 & m \neq \operatorname*{argmax}_m q_\star(s, a^m) \end{cases}$$

+ *But, they are both in terms of $q_\star(s, a^m)$! We don't have the optimal action-values!*
– Sure! But, we could say that optimal values must satisfy this constraint: *if not, they cannot be optimal*

Optimal Value: *Optimality Constraint*

Optimality Constraint

Optimal value at each state s satisfies the following identity

$$v_\star(s) = \max_m q_\star(s, a^m)$$

and is achieved if we set the policy to

$$\pi^\star(a^m|s) = \begin{cases} 1 & m = \operatorname*{argmax}_m q_\star(s, a^m) \\ 0 & m \neq \operatorname*{argmax}_m q_\star(s, a^m) \end{cases}$$

which is an optimal policy

+ *But, how can we relate this constraint to Bellman equation?*
- *Let's see!*

Optimal Value: *Bellman Equation*

We know from Bellman equation II for action-value function that

$$q_\pi(s,a) = \bar{\mathcal{R}}(s,a) + \gamma \sum_{n=1}^{N} v_\pi(s^n)\, p(s^n | s, a)$$

If we play with optimal policy: *we are going to have same identity*

$$q_\star(s,a) = \bar{\mathcal{R}}(s,a) + \gamma \sum_{n=1}^{N} v_\star(s^n)\, p(s^n | s, a)$$

We now substitute it in *optimality constraint*

$$v_\star(s) = \max_m \bar{\mathcal{R}}(s,a^m) + \gamma \sum_{n=1}^{N} v_\star(s^n)\, p(s^n | s, a^m)$$

Optimal Value: *Bellman Equation*

This is again a recursive equation that

does not depend on any policy!

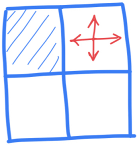
Bellman Optimality Equation

The optimal value function $v_\star(s)$ *satisfies*

$$v_\star(s) = \max_m \bar{\mathcal{R}}(s, a^m) + \gamma \sum_{n=1}^{N} v_\star(s^n)\, p(s^n | s, a^m)$$

We can again treat it as a fixed-point equation and solve it for $v_\star(s)$

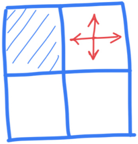
Example: *Dummy Grid World*



Let's find optimal values for our dummy grid world: we first find $\bar{\mathcal{R}}(s, a)$

$$
\begin{array}{llll}
\bar{\mathcal{R}}(0, a) = 0 & \bar{\mathcal{R}}(1, 0) = -1 & \bar{\mathcal{R}}(2, 0) = -0.5 & \bar{\mathcal{R}}(3, 0) = -1 \\
 & \bar{\mathcal{R}}(1, 1) = -1 & \bar{\mathcal{R}}(2, 1) = -0.5 & \bar{\mathcal{R}}(3, 1) = -0.5 \\
 & \bar{\mathcal{R}}(1, 2) = -0.5 & \bar{\mathcal{R}}(2, 2) = -1 & \bar{\mathcal{R}}(3, 2) = -0.5 \\
 & \bar{\mathcal{R}}(1, 3) = -0.5 & \bar{\mathcal{R}}(2, 3) = -1 & \bar{\mathcal{R}}(3, 3) = -1
\end{array}
$$

Example: *Dummy Grid World*

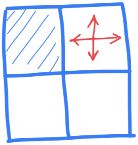


We next write down Bellman equations

1. *Since* $s = 0$ *is a terminal state we know that* $v_\star(0) = 0$
2. *Now, let's consider* $s = 1$

$$\begin{aligned} p(0|1,0) &= 1 \\ p(1|1,0) &= 0 \\ p(2|1,0) &= 0 \\ p(3|1,0) &= 0 \end{aligned} \rightsquigarrow \sum_{\bar{s}=0}^{4} v_\star(\bar{s})\, p(\bar{s}|1,0) = v_\star(0) = 0$$

Example: *Dummy Grid World*

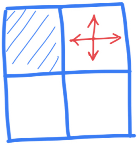


We next write down Bellman equations

1. *Since* $s = 0$ *is a terminal state we know that* $v_\star(0) = 0$
2. *Now, let's consider* $s = 1$

$$\begin{aligned} p(0|1,1) &= 0 \\ p(1|1,1) &= 0 \\ p(2|1,1) &= 0 \\ p(3|1,1) &= 1 \end{aligned} \rightsquigarrow \sum_{\bar{s}=0}^{4} v_\star(\bar{s})\, p(\bar{s}|1,1) = v_\star(3)$$

Example: *Dummy Grid World*

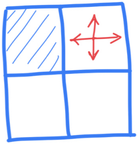


We next write down Bellman equations

1. *Since* $s = 0$ *is a terminal state we know that* $v_\star(0) = 0$
2. *Now, let's consider* $s = 1$

$$\begin{aligned} p(0|1,2) &= 0 \\ p(1|1,2) &= 1 \\ p(2|1,2) &= 0 \\ p(3|1,2) &= 0 \end{aligned} \rightsquigarrow \sum_{\bar{s}=0}^{4} v_\star(\bar{s})\, p(\bar{s}|1,2) = v_\star(1)$$

Example: *Dummy Grid World*

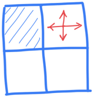


We next write down Bellman equations

1. *Since* $s = 0$ *is a terminal state we know that* $v_\star(0) = 0$
2. *Now, let's consider* $s = 1$

$$\begin{aligned} p(0|1,3) &= 0 \\ p(1|1,3) &= 1 \\ p(2|1,3) &= 0 \\ p(3|1,3) &= 0 \end{aligned} \rightsquigarrow \sum_{\bar{s}=0}^{4} v_\star(\bar{s})\, p(\bar{s}|1,3) = v_\star(1)$$

Example: *Dummy Grid World*

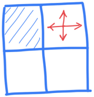


We next write down Bellman equations

1. *Since* $s = 0$ *is a terminal state we know that* $v_\star(0) = 0$
2. *Now, let's consider* $s = 1$

$$
\begin{aligned}
v_\star(1) &= \max_m \bar{\mathcal{R}}(1, a^m) + \sum_{\bar{s}=0}^{4} v_\star(\bar{s})\, p(\bar{s}|1, a^m) \\
&= \max\left\{-1, -1 + v_\star(3), -0.5 + v_\star(1), -0.5 + v_\star(1)\right\}
\end{aligned}
$$

Example: *Dummy Grid World*



We next write down Bellman equations

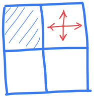
1. *Since* $s = 0$ *is a terminal state we know that* $v_\star(0) = 0$
2. *Now, let's consider* $s = 1$

$$v_\star(1) = \max\left\{-1, -1 + v_\star(3), -0.5 + v_\star(1), -0.5 + v_\star(1)\right\}$$

3. *Similarly, we have for* $s = 2$

$$v_\star(2) = \max\left\{-0.5 + v_\star(2), -0.5 + v_\star(2), -1 + v_\star(3), -1\right\}$$

Example: *Dummy Grid World*



We next write down Bellman equations

1. *Since* $s = 0$ *is a terminal state we know that* $v_\star(0) = 0$
2. *Now, let's consider* $s = 1$

$$v_\star(1) = \max\{-1, -1 + v_\star(3), -0.5 + v_\star(1), -0.5 + v_\star(1)\}$$

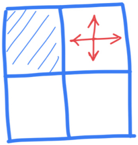
3. *Similarly, we have for* $s = 2$

$$v_\star(2) = \max\{-0.5 + v_\star(2), -0.5 + v_\star(2), -1 + v_\star(3), -1\}$$

4. *Finally for* $s = 3$*, we have*

$$v_\star(3) = \max\{-1 + v_\star(2), -0.5 + v_\star(3), -0.5 + v_\star(3), -1 + v_\star(1)\}$$

Example: *Dummy Grid World*

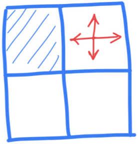


After sorting out the Bellman equations, we get

$$
\begin{aligned}
v_\star(1) &= \max\left\{-1, -1 + v_\star(3), -0.5 + v_\star(1)\right\}\\
v_\star(2) &= \max\left\{-1, -1 + v_\star(3), -0.5 + v_\star(2)\right\}\\
v_\star(3) &= \max\left\{-1 + v_\star(2), -0.5 + v_\star(3), -1 + v_\star(1)\right\}
\end{aligned}
$$

We should now solve this system of equations

Example: *Dummy Grid World*



We first note that

$$\max\left\{-1, -1 + v_\star\left(3\right), -0.5 + v_\star\left(1\right)\right\} \neq -0.5 + v_\star\left(1\right)$$

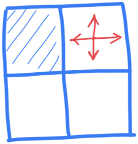
Proof: *Assume that*

$$\max\left\{-1, -1 + v_\star\left(3\right), -0.5 + v_\star\left(1\right)\right\} = -0.5 + v_\star\left(1\right)$$

Then, we have

$$v_\star\left(1\right) - 0.5 + v_\star\left(1\right) \rightsquigarrow 0 = -0.5 \qquad \text{impossible!}$$

Example: *Dummy Grid World*



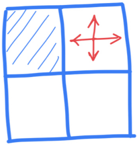
For the same reason, we have

$$\max \{-1, -1 + v_\star(3), -0.5 + v_\star(2)\} \neq -0.5 + v_\star(2)$$
$$\max \{-1 + v_\star(2), -0.5 + v_\star(3), -1 + v_\star(1)\} \neq -0.5 + v_\star(3)$$

So the equations reduce to

$$v_\star(1) = \max \{-1, -1 + v_\star(3)\} = v_\star(2)$$
$$v_\star(2) = \max \{-1, -1 + v_\star(3)\} = v_\star(1)$$
$$v_\star(3) = \max \{-1 + v_\star(2), -1 + v_\star(1)\} = -1 + v_\star(1)$$

Example: *Dummy Grid World*



Thus, we should only solve

$$v_\star(1) = \max\{-1, -1 + v_\star(3)\}$$
$$v_\star(3) = -1 + v_\star(1)$$

It is again easy to see that $\max\{-1, -1 + v_\star(3)\} \neq -1 + v_\star(3)$*; therefore,*

$$v_\star(1) = v_\star(2) = -1 \rightsquigarrow v_\star(3) = -2$$

Well! This is what we expected!

From Optimal Values to *Optimal Policy*

+ *What is the benefit then? It only finds optimal value, but we are looking for optimal policy!*
– We can actually back-track optimal policy, once we have optimal value

---

The idea is quite simple:

1. *We can find optimal values from Bellman optimality equations*
2. *We could then find the optimal action-values*
3. *We finally get the optimal policy from optimal action-values*

Finding Optimal Policy: *Back-Tracking from Optimal Values*

We could summarize this approach algorithmically as follows

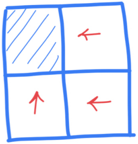
```
OptimBackTrack():
```

1: **for** $n = 1 : N$ **do**
2: *Solve Bellman equation* $v_\star(s^n) = \max_m \bar{\mathcal{R}}(s^n, a^m) + \gamma \mathbb{E}\left\{v_\star\left(\bar{S}\right) | s^n, a^m\right\}$
3: **end for**
4: **for** $n = 1 : N$ **do**
5: **for** $m = 1 : M$ **do**
6: *Compute action-value* $q_\star(s^n, a^m) = \bar{\mathcal{R}}(s^n, a^m) + \gamma \mathbb{E}\left\{v_\star\left(\bar{S}\right) | s^n, a^m\right\}$
7: **end for**
8: *Compute optimal policy via optimality constraint*

$$\pi^\star(a^m|s) = \begin{cases} 1 & m = \operatorname*{argmax}_m q_\star(s, a^m) \\ 0 & m \neq \operatorname*{argmax}_m q_\star(s, a^m) \end{cases}$$

9: **end for**

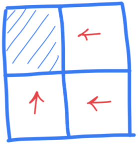
Example: *Dummy Grid World*



Let's find optimal policy at state $s = 1$ in our dummy grid world: first, we write

$$\begin{bmatrix} q_\star(1,0) \\ q_\star(1,1) \\ q_\star(1,2) \\ q_\star(1,3) \end{bmatrix} = \begin{bmatrix} \bar{\mathcal{R}}(1,0) + \sum_{\bar{s}} v_\star(\bar{s})\, p(\bar{s}|1,0) \\ \bar{\mathcal{R}}(1,1) + \sum_{\bar{s}} v_\star(\bar{s})\, p(\bar{s}|1,1) \\ \bar{\mathcal{R}}(1,2) + \sum_{\bar{s}} v_\star(\bar{s})\, p(\bar{s}|1,2) \\ \bar{\mathcal{R}}(1,3) + \sum_{\bar{s}} v_\star(\bar{s})\, p(\bar{s}|1,3) \end{bmatrix} = \begin{bmatrix} -1+0 \\ -1-2 \\ -0.5-1 \\ -0.5-1 \end{bmatrix} = \begin{bmatrix} -1 \\ -3 \\ -1.5 \\ -1.5 \end{bmatrix}$$

Example: *Dummy Grid World*



The optimal policy at state $s = 1$ is then given by

$$\pi^\star(a|1) = \begin{cases} 1 & a = \operatorname*{argmax}_a q_\star(1, a) \\ 0 & a \neq \operatorname*{argmax}_a q_\star(1, a) \end{cases} = \begin{cases} 1 & a = 0 \\ 0 & a \neq 0 \end{cases}$$

Well! We know that this is optimal in this problem!

Finding Optimal Policy: *Back-Tracking from Optimal Values*

+ *Wait a moment! Does that mean that our optimal policy is always deterministic? But, you said it could be also random!*
– Well! In some cases we could find random optimal policies as well!

If $q_\star(s, a^m)$ has a single maximizer; then,

optimal policy $\pi^\star(a^m|s)$ is deterministic

But, if it has multiple maximizers

optimal policy $\pi^\star(a^m|s)$ can also be random

Finding Optimal Policy: *General Form*

Generic Optimal Policy

Assume that $m^1, \ldots, m^J$ are all maximizers of $q_\star(s, a^m)$; then, policy

$$\pi^\star(a^m|s) = \begin{cases} p_1 & m = m^1 \\ \vdots & \\ p_J & m = m^J \\ 0 & m \notin \{m^1, \ldots, m^J\} \end{cases}$$

for any $p_1, \ldots, p_J$ that satisfy

$$\sum_{j=1}^{J} p_j = 1$$

is an optimal policy

Finding Optimal Policy

+ *But, why are all such policies optimal?*
- Well! We could look back at the optimality constraint

With any policy $\pi^\star(a|s)$ of the form given in the last slide, we have

$$v_{\pi^\star}(s) = \sum_{m=1}^{M} \pi^\star(a^m|s)\, q_{\pi^\star}(s, a^m) = \sum_{j=1}^{J} p_j q_{\pi^\star}\left(s, a^{m^j}\right) + 0$$

$$= \sum_{j=1}^{J} p_j \max_m q_{\pi^\star}(s, a^m) = \max_m q_{\pi^\star}(s, a^m) \sum_{j=1}^{J} p_j = \max_m q_{\pi^\star}(s, a^m)$$

which is the optimality constraint! It's intuitive, because

> *If we have multiple options for next action that give us same maximal value; then, we could randomly pick any of them*

Finding Optimal Policy

+ *But, still we could have a deterministic optimal policy in such cases! Right?!*
– Sure! *We could always have a deterministic optimal policy!*

Deterministic Optimal Policy

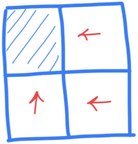
With known MDP for the environment, there exists at least one deterministic optimal policy

---

In the nutshell: if we know the complete state and its transition model

- *We always can find a deterministic optimal policy*
- *We might have multiple deterministic optimal policies*
  ↳ *In that case, we are going to have also random optimal policies*

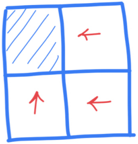
Example: *Dummy Grid World*



Let's find optimal policy at state $s = 3$ in our dummy grid world: first, we write

$$\begin{bmatrix} q_\star(3,0) \\ q_\star(3,1) \\ q_\star(3,2) \\ q_\star(3,3) \end{bmatrix} = \begin{bmatrix} \bar{\mathcal{R}}(3,0) + \sum_{\bar{s}} v_\star(\bar{s})\, p(\bar{s}|3,0) \\ \bar{\mathcal{R}}(3,1) + \sum_{\bar{s}} v_\star(\bar{s})\, p(\bar{s}|3,1) \\ \bar{\mathcal{R}}(3,2) + \sum_{\bar{s}} v_\star(\bar{s})\, p(\bar{s}|3,2) \\ \bar{\mathcal{R}}(3,3) + \sum_{\bar{s}} v_\star(\bar{s})\, p(\bar{s}|3,3) \end{bmatrix} = \begin{bmatrix} -1-1 \\ -0.5-2 \\ -0.5-2 \\ -1-1 \end{bmatrix} = \begin{bmatrix} -2 \\ -2.5 \\ -2.5 \\ -2 \end{bmatrix}$$

Example: *Dummy Grid World*

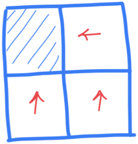


The optimal policy at state $s = 3$ is then given by

$$\pi^\star(a|3) = \begin{cases} 1 & a = \operatorname*{argmax}_a q_\star(3, a) \\ 0 & a \neq \operatorname*{argmax}_a q_\star(3, a) \end{cases} = \begin{cases} 1 & a = 0 \\ 0 & a \neq 0 \end{cases}$$

This is obviously optimal in this problem!

Example: *Dummy Grid World*

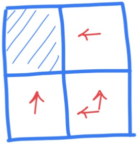


The optimal policy at state $s = 3$ is then given by

$$\pi^\star (a|3) = \begin{cases} 1 & a = \operatorname*{argmax}_a q_\star (3, a) \\ 0 & a \neq \operatorname*{argmax}_a q_\star (3, a) \end{cases} = \begin{cases} 1 & a = 3 \\ 0 & a \neq 3 \end{cases}$$

This is obviously optimal in this problem!

Example: *Dummy Grid World*

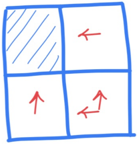


The optimal policy at state $s = 3$ is then given by

$$\pi^\star(a|3) = \begin{cases} 0.5 & a = 0 \\ 0 & a = 1, 2 \\ 0.5 & a = 3 \end{cases}$$

This is also optimal in this problem!

Example: *Dummy Grid World*

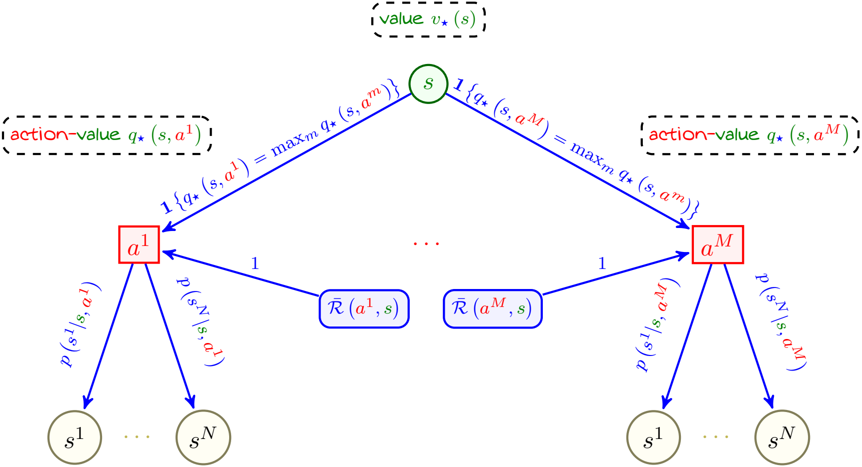


The optimal policy at state $s = 3$ is then given by

$$\pi^\star(a|3) = \begin{cases} 0.2 & a = 0 \\ 0 & a = 1, 2 \\ 0.8 & a = 3 \end{cases}$$

This is also optimal in this problem!

Backup Diagram: *For Optimal Policy*



Here, we assume $q_\star(s, a^m)$ *has one maximizer* $\equiv$ *optimal policy is deterministic*

Last Piece: *Dynamic Programming*

Right now, we know what to do *when we know MDP of environment*

1. *We can find optimal values from Bellman optimality equations*
2. *We could then find the optimal action-values*
3. *We finally get the optimal policy from optimal action-values*

The only remaining challenge is to find

an algorithmic approach to solve Bellman optimality equations

We complete this last piece using

Dynamic Programming $\equiv$ *DP*

Dynamic Programming: *Basic Idea*

Assume, we want to solve the problem of

$$x = f(x)$$

for some function $f(x)$

We could solve it via direct approach:

1. *Rewrite is as $f(x) - x = 0$*
2. *Solve it via classic algorithms*
   - ↳ *Reduce it to a known form, e.g., a polynomial*
   - ↳ *Solve it via an iterative method, e.g., Newton-Raphson or method of intervals*

Dynamic Programming: *Basic Idea*

Assume, we want to solve the problem of

$$x = f(x)$$

for some function $f(x)$

We could also solve it by recursion:

1. *Start with an x^0 and set $x^1 = f(x^0)$*
2. *Until $x^{k+1} \approx x^k$, we do*
    - ↳ *Update recursively as $x^{k+1} = f(x^k)$*
    - ↳ *Set $k \leftarrow k + 1$*

Under some conditions on $f(\cdot)$, this approach can converge

Dynamic Programming: *Example*

We want to solve

$$x = \frac{-1}{2+x}$$

1. *Start with an* $x^0 = 0$
2. *We now get into the recursion loop*
   - ↳ $x^1 = f\left(x^0\right) = -\frac{1}{2}$
   - ↳ $x^2 = f\left(x^1\right) = -\frac{2}{3}$
   - ↳ $x^3 = f\left(x^2\right) = -\frac{3}{4}$
   - ↳ $\cdots$
   - ↳ $x^k = f\left(x^{k-1}\right) = -\frac{k}{k+1}$

We asymptotically converge to $x^\infty = -1$ which is the solution

↳ *Note that we always converge no matter which point we start*

Dynamic Programming: *Example*

Now, let's write the *same equation* in a *different recursive form*

$$x = \frac{-1 - x^2}{2}$$

1. *Start with an* $x^0 = 0$
2. *We get into recursion loop*
   - ↳ $x^1 = f\left(x^0\right) = -0.5$
   - ↳ $x^2 = f\left(x^1\right) = -0.625$
   - ↳ $\cdots$
   - ↳ $x^\infty = -1$

1. *Start with an* $x^0 = 5$
2. *We get into recursion loop*
   - ↳ $x^1 = f\left(x^0\right) = -13$
   - ↳ $x^2 = f\left(x^1\right) = -85$
   - ↳ $\cdots$
   - ↳ $x^\infty = -\infty$

We can now diverge if we start with a wrong initial point!

Not all recursive forms are always converging!

Dynamic Programming: *Applications to Our Problem*

Our problem has a similar form: *we need to solve Bellman equations*

which are recursive equations

So, we could use DP to find the solution

There are two major DP approaches

- *Policy Iteration that uses recursion to iterate between*
  - ↳ *Policy Evaluation*
  - ↳ *Policy Improvement*
- *Value Iteration which applies recursion on optimal Bellman equation*

Let's look at these two approaches in detail

Policy Evaluation: *Step I*

The first step is *policy evaluation*: *we can formulate this problem as follows*

Ultimate Goal of Policy Evaluation

Given a policy π, we intend to evaluate values of all states by recursion

---

Before we start, let's recap a few definitions: recall expected policy reward

$$\bar{\mathcal{R}}_\pi(s) = \sum_{m=1}^{M} \bar{\mathcal{R}}(a^m, s)\, \pi(a^m|s)$$

For sake of compactness, we use the following notation

$$\bar{\mathcal{R}}_\pi(s) = \mathbb{E}_\pi\left\{\bar{\mathcal{R}}(A, s)\,|s\right\}$$

Policy Evaluation: *Step I*

Similarly, we define the notation

$$\mathbb{E}_{\pi}\left\{v_{\pi}\left(\bar{S}\right)|s,a\right\} = \sum_{n=1}^{N} v_{\pi}\left(s^{n}\right) p\left(s^{n}|s,a\right)$$

and also denote its expected form over the action set by

$$\begin{aligned}\mathbb{E}_{\pi}\left\{v_{\pi}\left(\bar{S}\right)|s\right\} &= \sum_{n=1}^{N} v_{\pi}\left(s^{n}\right) p_{\pi}\left(s^{n}|s\right) \\ &= \sum_{m=1}^{M} \underbrace{\sum_{n=1}^{N} v_{\pi}\left(s^{n}\right) p\left(s^{n}|s,a^{m}\right)} \pi\left(a^{m}|s\right) \\ &= \sum_{m=1}^{M} \mathbb{E}_{\pi}\left\{v_{\pi}\left(\bar{S}\right)|s,a^{m}\right\} \pi\left(a^{m}|s\right)\end{aligned}$$

Policy Evaluation: *Step I*

We can then write the Bellman equations compactly as

$$v_\pi(s) = \bar{\mathcal{R}}_\pi(s) + \gamma \mathbb{E}_\pi\left\{ v_\pi\left(\bar{S}\right) | s \right\}$$

for value function and also as

$$q_\pi(s, a) = \bar{\mathcal{R}}(s, a) + \gamma \mathbb{E}_\pi\left\{ v_\pi\left(\bar{S}\right) | s, a \right\}$$

for action-value function

Now, we are ready to evaluate a policy by recursion

Policy Evaluation: *Value Computation via Recursion*

Recall our perspective on value computation:

values are N unknowns that we want to compute from Bellman equations

Now, if someone claims that *the values*

$$v_\pi(s^n) = v_n$$

for $n = 1 : N$ are values of policy π, can we confirm it?

+ *Shouldn't we simply use Bellman Equation?!*
– Exactly!

Policy Evaluation: *Value Computation via Recursion*

We could confirm

$$v_\pi\left(s^n\right) = v_n$$

by writing first finding for every state s

$$\begin{aligned}
\mathbb{E}_\pi\left\{v_\pi\left(\bar{S}\right)|s\right\} &= \sum_{n=1}^{N} v_\pi\left(s^n\right) p_\pi\left(s^n|s\right) \\
&= \sum_{n=1}^{N}\sum_{m=1}^{M} v_\pi\left(s^n\right) p\left(s^n|s,a^m\right) \pi\left(a^m|s\right) \\
&= \sum_{n=1}^{N}\sum_{m=1}^{M} \underbrace{v_n}_{\text{claimed value}} \underbrace{p\left(s^n|s,a^m\right)}_{\text{transition model}} \underbrace{\pi\left(a^m|s\right)}_{\text{policy}}
\end{aligned}$$

Policy Evaluation: *Value Computation via Recursion*

We could confirm

$$v_\pi(s^n) = v_n$$

by writing first finding for every state s

$$\mathbb{E}_\pi\left\{v_\pi\left(\bar{S}\right)|s\right\} = \text{computed from } v_n\text{'s} := F\left(\{v_1, \ldots, v_N\}, s\right)$$

and then checking if

$$\begin{aligned} v_\pi(s^n) = v_n &= \bar{\mathcal{R}}_\pi(s^n) + \gamma \mathbb{E}_\pi\left\{v_\pi\left(\bar{S}\right)|s^n\right\} \\ &= \bar{\mathcal{R}}_\pi(s^n) + \gamma F\left(\{v_1, \ldots, v_N\}, s\right) \end{aligned}$$

holds for all $n = 1 : N$

Policy Evaluation: *Value Computation via Recursion*

If it happens that the claimed $v_\pi(\cdot)$ is not a valid claim; then, *we get out of Bellman equation*

$$\bar{v}_\pi(s^n) = \bar{v}_n = \bar{\mathcal{R}}_\pi(s^n) + \gamma \mathbb{E}_\pi\left\{ v_\pi\left(\bar{S}\right) | s^n \right\}$$

which is different from the claimed $v_\pi(\cdot)$, i.e., $v_n \neq \bar{v}_n$

Policy Evaluation

We iterate this procedure until we can confirm, i.e., we

1. *set $v_\pi(\cdot) \leftarrow \bar{v}_\pi(\cdot)$*
2. *repeat the same procedure and compute new $\bar{v}_\pi(\cdot)$*

We stop when $v_\pi(\cdot) = \bar{v}_\pi(\cdot)$, or at least it happens approximately

Policy Evaluation

`PolicyEval` $\left(\pi, v_\pi^0\right)$:

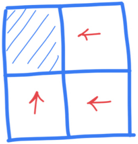
Initiate values with v_π^0 *and set* $k = 0$
Make sure that $v_\pi^0\left(s\right) = 0$ *for* ***terminal*** *states* s
Choose a small threshold ϵ *and initiate* $\Delta = +\infty$ `# stopping criteria`
for $n = 1 : N$ **do**
  Compute $\bar{\mathcal{R}}_\pi\left(s^n\right) = \mathbb{E}_\pi\left\{\bar{\mathcal{R}}\left(s^n, a\right)\right\}$ `# average rewards`
end for
while $\Delta > \epsilon$ **do**
  for $n = 1 : N$ **do**
    Update $v_\pi^{k+1}\left(s^n\right) = \bar{\mathcal{R}}_\pi\left(s^n\right) + \gamma \mathbb{E}_\pi\left\{v_\pi^k\left(\bar{S}\right) | s^n\right\}$ `# DP update`
  end for
  $\Delta = \max_n \left| v_\pi^{k+1}\left(s^n\right) - v_\pi^k\left(s^n\right)\right|$ `# check convergence`
  Update $k \leftarrow k + 1$
end while

Recursion Loop

Attention

We should make sure that terminal states are all initiated with zero value

Example: *Dummy Grid World*



Let's try with our dummy grid world: we saw that

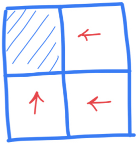
$$\bar{\mathcal{R}}_\pi(0) = 0 \qquad \bar{\mathcal{R}}_\pi(1) = -1 \qquad \bar{\mathcal{R}}_\pi(2) = -1 \qquad \bar{\mathcal{R}}_\pi(3) = -1$$

Now let's evaluate its values by recursion: we first note that, if we have

$$\mathbb{E}_\pi\left\{v_\pi^k\left(\bar{S}\right)|0\right\} = v_\pi^k(0) \qquad \mathbb{E}_\pi\left\{v_\pi^k\left(\bar{S}\right)|1\right\} = v_\pi^k(0)$$

$$\mathbb{E}_\pi\left\{v_\pi^k\left(\bar{S}\right)|2\right\} = v_\pi^k(0) \qquad \mathbb{E}_\pi\left\{v_\pi^k\left(\bar{S}\right)|3\right\} = v_\pi^k(2)$$

Example: *Dummy Grid World*



$\texttt{PolicyEval}(\pi, v_\pi^0)$:

Initiate values with $v_\pi^0(1)$, $v_\pi^0(2)$ *and* $v_\pi^0(3)$ *at random and set* $v_\pi^0(0) = 0$
Set $\epsilon = 0.001$, *and initiate* $\Delta = 1000$ `# stopping criteria`
while $\Delta > \epsilon$ ***do***
  Update $v_\pi^{k+1}(1) = -1 + v_\pi^k(0)$ `# DP update`
  Update $v_\pi^{k+1}(2) = -1 + v_\pi^k(0)$ `# DP update`
  Update $v_\pi^{k+1}(3) = -1 + v_\pi^k(2)$ `# DP update`
  $\Delta = \max_{s\in\{1,2,3\}} |v_\pi^{k+1}(s) - v_\pi^k(s)|$ `# check convergence`
  Update $k \leftarrow k + 1$
end while

It converges after only one recursion!

Policy Improvement

Let us know recall optimality constraint: *with optimal policy, we have*

$$v_\star(s) = \max_m q_\star(s, a^m)$$

which can be achieved by policy

$$\pi^\star(a^m|s) = \begin{cases} 1 & m = \operatorname*{argmax}_m q_\star(s, a^m) \\ 0 & m \neq \operatorname*{argmax}_m q_\star(s, a^m) \end{cases}$$

This means that if π is not optimal, we would have

$$\pi(a^m|s) \neq \begin{cases} 1 & m = \operatorname*{argmax}_m q_\pi(s, a^m) \\ 0 & m \neq \operatorname*{argmax}_m q_\pi(s, a^m) \end{cases}$$

Policy Improvement

In other words, if we change our policy to

$$\bar{\pi}\left(a^m|s\right) = \begin{cases} 1 & m = \underset{m}{\operatorname{argmax}}\, q_\pi\left(s, a^m\right) \\ 0 & m \neq \underset{m}{\operatorname{argmax}}\, q_\pi\left(s, a^m\right) \end{cases}$$

Then, it should give us better values, i.e., $\bar{\pi} \geqslant \pi$!

+ *Are you sure?! I don't see it immediately*
- We can actually show it!

This is what we call policy improvement theorem

Policy Improvement

Policy Improvement

Given (deterministic) policy π^k, we can always design a better policy π^{k+1} by setting it to

$$\pi^{k+1}(a^m|s) = \begin{cases} 1 & m = \underset{m}{\operatorname{argmax}}\, q_{\pi^k}(s, a^m) \\ 0 & m \neq \underset{m}{\operatorname{argmax}}\, q_{\pi^k}(s, a^m) \end{cases}$$

Policy Improvement

```
PolicyImprov(v_π):
1: for n = 1 : N do
2:     for m = 1 : M do
3:         Compute R̄(s^n, a^m)
4:         q_π(s^n, a^m) = R̄(s^n, a^m) + γE_π{v_π(S̄)|s^n, a^m}     # action-values
5:     end for
6:     Compute an improved policy as                                 # policy improvement
```

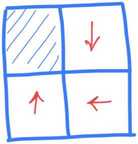
$$\bar{\pi}\left(a^m | s^n\right) = \begin{cases} 1 & m = \underset{m}{\operatorname{argmax}}\, q_\pi\left(s^n, a^m\right) \\ 0 & m \neq \underset{m}{\operatorname{argmax}}\, q_\pi\left(s^n, a^m\right) \end{cases}$$

```
7: end for
```

Attention

Here, we do no recursion

Example: *Dummy Grid World*



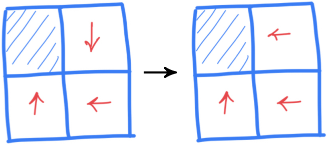
Let's try dummy grid world with above non-optimal policy: here, we have

$$v_\pi(0) = 0 \qquad v_\pi(1) = -3 \qquad v_\pi(2) = -1 \qquad v_\pi(3) = -2$$

We now look at acion-values at the problematic state $s = 1$

$$\begin{aligned} q_\pi(1, 0) &= -1 \\ q_\pi(1, 1) &= -3 \\ q_\pi(1, 2) &= -3.5 \\ q_\pi(1, 3) &= -3.5 \end{aligned} \rightsquigarrow -3 = v_\pi(1) \neq \max_a q_\pi(1, a) = -1$$

Example: *Dummy Grid World*



Now if we improve the policy, we get

$$\bar{\pi}(a|1) = \begin{cases} 1 & m = \operatorname*{argmax}_m q_\pi(1, a) \\ 0 & m \neq \operatorname*{argmax}_m q_\pi(1, a) \end{cases} = \begin{cases} 1 & a = 0 \\ 0 & a \neq 0 \end{cases}$$

which is actually optimal

Policy Iteration: *Improving Policy by Recursion*

Looking at the policy improvement theorem, we see

> *If we plug in $\pi^k = \pi^\star$ into algorithm; then, after policy improvement*
>
> $\hookrightarrow$ *we get $\pi^{k+1} = \pi^\star$*
>
> $\hookrightarrow$ *say we evaluate values for $\pi^{k+1} = \pi^\star$ and plug back to algorithm*
>
> $\quad\hookrightarrow$ *we get $\pi^{k+2} = \pi^\star$*
>
> $\quad\hookrightarrow$ *say we evaluate values for $\pi^{k+2} = \pi^\star$ and plug back to algorithm*
>
> $\qquad\hookrightarrow$ $\ldots$

So, optimal policy is a fixed-point for this recursion

Policy Iteration

We can start with an arbitrary policy π^0 and keep doing the above recursion until we see that $\pi^{k+1} = \pi^k$ which indicates that we reached optimal policy

Policy Iteration

```
PolicyItr():
```

1: *Initiate with random* $v_\pi(s)$ *for all* ***non-terminal*** *states* s
2: *Set* $v_\pi(s) = 0$ *for* ***terminal*** *states* s
3: *Initiate two random policies* π *and* $\bar{\pi}$
4: ***while*** $\pi \neq \bar{\pi}$ ***do***
5: $v_\pi = \texttt{PolicyEval}(\pi, v_\pi)$ *and* $\pi \leftarrow \bar{\pi}$ Recursion
6: $\bar{\pi} = \texttt{PolicyImprov}(v_\pi)$
7: ***end while*** Recursion

Note that this is a *nested recursive computation*

- *There is a loop for recursion inside the algorithm in which*
  - ↳ *at each iteration we evaluate the policy recursively*
- *But, we initiate each policy evaluation loop with the values of last iteration*
  - ↳ *this can improve the convergence speed*

Back-Tracking by Recursion

+ *But wait a Moment! We already talked about back-tracking optimal policy from Bellman optimality equation! Don't we implement that?!*
– Sure! We can do the same thing by recursion

We follow the same idea but we use recursion

1. *We can find optimal values from Bellman optimality equations*
   ↳ *This is where we use recursion*
2. *We could then find the optimal action-values*
3. *We finally get the optimal policy from optimal action-values*

Recall: *Back-Tracking from Optimal Values*

```
OptimBackTrack():
```

1: *Solve Bellman equations* `# we use recursion`
2: **for** $n = 1 : N$ **do**
3: **for** $m = 1 : M$ **do**
4: Set $q_\star(s^n, a^m) = \bar{\mathcal{R}}(s^n, a^m) + \gamma \mathbb{E}\left\{v_\star\left(\bar{S}\right) | s^n, a^m\right\}$ `# action-values`
5: **end for**
6: Compute optimal policy via optimality constraint

$$\pi^\star(a^m|s) = \begin{cases} 1 & m = \operatorname*{argmax}_m q_\star(s, a^m) \\ 0 & m \neq \operatorname*{argmax}_m q_\star(s, a^m) \end{cases}$$

7: **end for**

Recursion with Bellman Optimality

Recall Bellman optimality equation

$$v_\star(s) = \max_m \left(\bar{\mathcal{R}}(s, a^m) + \gamma \mathbb{E}\left\{ v_\star(\bar{S}) \,|s, a^m \right\} \right)$$

We can again solve it by recursion: *we start with some* $v_\star^0(\cdot)$ *and then for every state* s *and action* a^m*, we compute*

$$\mathbb{E}\left\{ v_\star^k(\bar{S}) \,|s, a^m \right\} = \sum_{n=1}^{N} \underbrace{v_\star^k(s^n)}_{\text{last computed value}} \underbrace{p(s^n|s, a^m)}_{\text{transition model}}$$

We then update the optimal value function as

$$v_\star^{k+1}(s) = \max_m \left(\bar{\mathcal{R}}(s, a^m) + \gamma \mathbb{E}\left\{ v_\star^k(\bar{S}) \,|s, a^m \right\} \right)$$

Value Iteration vs *Policy Iteration*

Before we complete the value iteration algorithm: *it is interesting to put its recursion next to the one used for policy evaluation*

With optimality equation, we iterate as

$$v_\star^{k+1}(s) = \max_m \left(\bar{\mathcal{R}}(s, a^m) + \gamma \mathbb{E}\left\{ v_\star^k(\bar{S}) \,|s, a^m \right\} \right)$$

With Bellman equation for a given policy π, we iterate as

$$\begin{aligned} v_\pi^{k+1}(s) &= \bar{\mathcal{R}}_\pi(s) + \gamma \mathbb{E}_\pi \left\{ v_\pi^k(\bar{S}) \,|s \right\} \\ &= \sum_{m=1}^{M} \left(\bar{\mathcal{R}}(s, a^m) + \gamma \mathbb{E}\left\{ v_\pi^k(\bar{S}) \,|s, a^m \right\} \right) \pi(a^m|s) \end{aligned}$$

Value Iteration vs *Policy Iteration*

With optimality equation, we iterate as

$$v_\star^{k+1}(s) = \max_m \left(\bar{\mathcal{R}}(s, a^m) + \gamma \mathbb{E}\left\{ v_\star^k(\bar{S}) \,|s, a^m \right\} \right)$$

With Bellman equation for a given policy π, we iterate as

$$v_\pi^{k+1}(s) = \sum_{m=1}^{M} \left(\bar{\mathcal{R}}(s, a^m) + \gamma \mathbb{E}\left\{ v_\pi^k(\bar{S}) \,|s, a^m \right\} \right) \pi(a^m|s)$$

This indicates that for both recursive loops

- *we compute M values per iteration per state*
  - ↳ *in policy iteration, we compute the average of these M via π*
  - ↳ *in value iteration, we take the largest among these M values*

Value Iteration

`ValueItr():`

Initiate with random $v_\star^0(s)$ *for all states, and set* $v_\star^0(s) = 0$ *for* ***terminal*** *states*

Choose a small threshold ϵ, *initiate* $\Delta = +\infty$ *and* $k = 0$

while $\Delta > \epsilon$ ***do***

    for $n = 1 : N$ ***do***

        for $m = 1 : M$ ***do***

            Compute $q_\star(s^n, a^m) = \bar{\mathcal{R}}(s^n, a^m) + \gamma \mathbb{E}\left\{ v_\star^k(\bar{S}) \,|s^n, a^m \right\}$

        end for

        Update $v_\pi^{k+1}(s^n) = \max_m q_\star(s^n, a^m)$ `# DP update`

    end for

    Set $\Delta = \max_n |v_\pi^{k+1}(s^n) - v_\pi^k(s^n)|$ *and* $k \leftarrow k + 1$

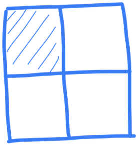
end while

Recursion

Compute an optimal policy as

$$\bar{\pi}(a^m|s) = \begin{cases} 1 & m = \operatorname*{argmax}_m q_\star(s, a^m) \\ 0 & m \neq \operatorname*{argmax}_m q_\star(s, a^m) \end{cases}$$

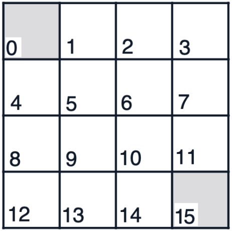
Example: *Dummy Grid World*



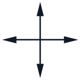
You may try policy and value iteration for this problem at home!

Easy as Pie ☺

Example: *A Bit Larger Grid World*[1]



Board ≡ states



moves ≡ actions

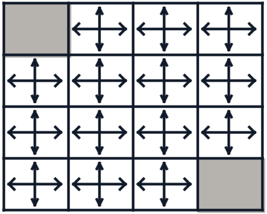
Let's do a bit of more serious example: we are now in a 4×4 grid world

- *We have two terminal states shown in gray*
- *Each move we do gets a -1 reward*
  - ↳ *We also get -1 reward if we hit a corner*
  - ↳ *We get zero reward at terminal state*

In simple words: we are looking for shortest path to the corners

[1] *This example is taken from Sutton and Barto's Book; Example 4.1 in Chapter 4*

Example: *A Bit Larger Grid World*



initial policy

0.0	0.0	0.0	0.0
0.0	0.0	0.0	0.0
0.0	0.0	0.0	0.0
0.0	0.0	0.0	0.0

initial values

Let's first try policy iteration: we start with

- *a uniform random policy* π^0
- *all values being zero, i.e.,* $v_{\pi^0}^0(s) = 0$ *for all* s

Example: *A Bit Larger Grid World*

Recall policy iteration:

```
PolicyItr():
```

1: *Initiate with random* $v_\pi(s)$ *for all* ***non-terminal*** *states* s
2: *Set* $v_\pi(s) = 0$ *for* ***terminal*** *states* s
3: *Initiate two random policies* π *and* $\bar{\pi}$
4: ***while*** $\pi \neq \bar{\pi}$ ***do***
5: $\quad v_\pi = \mathtt{PolicyEval}(\pi, v_\pi)$ *and* $\pi \leftarrow \bar{\pi}$ Recursion
6: $\quad \bar{\pi} = \mathtt{PolicyImprov}(v_\pi)$
7: ***end while***

Recursion

We should start with $v^0_{\pi^0}(\cdot)$ *and do the red recusion first*

- *at the end of this recursion we have evaluated the random policy*

Example: *A Bit Larger Grid World*

0.0	0.0	0.0	0.0
0.0	0.0	0.0	0.0
0.0	0.0	0.0	0.0
0.0	0.0	0.0	0.0

$v_{\pi^0}^0$

0.0	-1.0	-1.0	-1.0
-1.0	-1.0	-1.0	-1.0
-1.0	-1.0	-1.0	-1.0
-1.0	-1.0	-1.0	0.0

$v_{\pi^0}^1$

0.0	-1.7	-2.0	-2.0
-1.7	-2.0	-2.0	-2.0
-2.0	-2.0	-2.0	-1.7
-2.0	-2.0	-1.7	0.0

$v_{\pi^0}^2$

0.0	-2.4	-2.9	-3.0
-2.4	-2.9	-3.0	-2.9
-2.9	-3.0	-2.9	-2.4
-3.0	-2.9	-2.4	0.0

$v_{\pi^0}^3$

$\cdots$

0.0	-14.	-20.	-22.
-14.	-18.	-20.	-20.
-20.	-20.	-18.	-14.
-22.	-20.	-14.	0.0

$v_{\pi^0}^\infty$

We now have evaluated the value of random policy $v_{\pi^0} = v_{\pi^0}^\infty$

Example: *A Bit Larger Grid World*

Recall policy iteration:

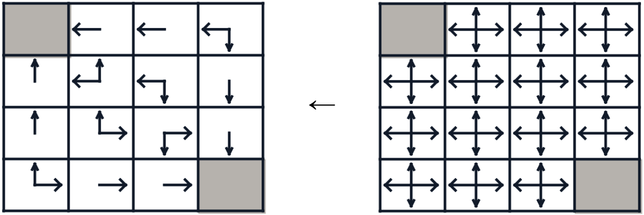
```
PolicyItr():
```

1: *Initiate with random $v_\pi(s)$ for all **non-terminal** states s*
2: *Set $v_\pi(s) = 0$ for **terminal** states s*
3: *Initiate two random policies π and $\bar{\pi}$*
4: ***while** $\pi \neq \bar{\pi}$ **do***
5: $\quad v_\pi = \texttt{PolicyEval}(\pi, v_\pi)$ *and* $\pi \leftarrow \bar{\pi}$ Recursion
6: $\quad \bar{\pi} = \texttt{PolicyImprov}(v_\pi)$
7: ***end while*** Recursion

Next, we do the outer recusion recursion, i.e.,

- *we improve the policy*

Example: *A Bit Larger Grid World*



π^1 $\longleftarrow$ π^0

We improve policy by taking actions with maximal action-values

- *if we have multiple maximal action-values we can behave randomly*

Example: *A Bit Larger Grid World*

Recall policy iteration:

```
PolicyItr():
```
1: *Initiate with random* $v_\pi(s)$ *for all* ***non-terminal*** *states* s
2: *Set* $v_\pi(s) = 0$ *for* ***terminal*** *states* s
3: *Initiate two random policies* π *and* $\bar{\pi}$
4: ***while*** $\pi \neq \bar{\pi}$ ***do***
5: $v_\pi = \mathtt{PolicyEval}(\pi, v_\pi)$ *and* $\pi \leftarrow \bar{\pi}$ Recursion
6: $\bar{\pi} = \mathtt{PolicyImprov}(v_\pi)$
7: ***end while*** Recursion

We now $\boxed{\textit{start with } v_{\pi^1}^0 = v_{\pi^0} = v_{\pi^0}^\infty}$ *and do the red recusion again*

- *at the end of this recursion we have evaluated the new policy* π^1

Example: *A Bit Larger Grid World*

0.0	-14.	-20.	-22.
-14.	-18.	-20.	-20.
-20.	-20.	-18.	-14.
-22.	-20.	-14.	0.0

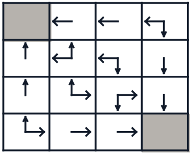
$v_{\pi^1}^{0}$

$\cdots$

0.0	-1.0	-2.0	-3.0
-1.0	-2.0	-3.0	-2.0
-2.0	-3.0	-2.0	-1.0
-3.0	-2.0	-1.0	0.0

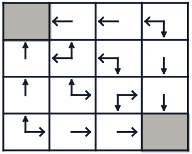
$v_{\pi^1}^{+\infty}$

After evaluating policy π^1 as $v_{\pi^1} = v_{\pi^1}^{\infty}$, we do the next improvement



$\pi^2 = \pi^1$

$\longleftarrow$



π^1

Well $\pi^2 = \pi^1$ and we should stop!

Example: *A Bit Larger Grid World*

0.0	0.0	0.0	0.0
0.0	0.0	0.0	0.0
0.0	0.0	0.0	0.0
0.0	0.0	0.0	0.0

initial values

Now we try value iteration: for start, we only need an initial value, so we set

- *all values being zero, i.e.,* $v_\star^0(s) = 0$ *for all* s

We keep recursion until we find the optimal values

Example: *A Bit Larger Grid World*

0.0	0.0	0.0	0.0
0.0	0.0	0.0	0.0
0.0	0.0	0.0	0.0
0.0	0.0	0.0	0.0

$v_\star^0$

$\cdots$

0.0	-1.0	-2.0	-3.0
-1.0	-2.0	-3.0	-2.0
-2.0	-3.0	-2.0	-1.0
-3.0	-2.0	-1.0	0.0

$v_\star^{+\infty}$

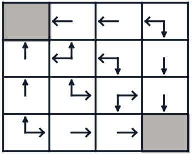
Now, we back-track the optimal policy $\pi^\star$

0.0	-1.0	-2.0	-3.0
-1.0	-2.0	-3.0	-2.0
-2.0	-3.0	-2.0	-1.0
-3.0	-2.0	-1.0	0.0

$v_\star$

$\longrightarrow$ action-values $\longrightarrow$

$q_\star$



$\pi^\star$

Complexity of Policy and Value Iteration

+ *It seems that value iteration has less complexity!*
- Well, it is not in order, but yes! It usually converge faster

---

In our example with policy iteration, we had to evaluate two policies

- *once for π^0 and once for π^1*
- *say the first recursion took K_1 iterations and the second took K_2*
  - ↳ *the total number of iterations is then $K_1 + K_2$*
  - ↳ *in practice, it often happens that $K_2 \ll K_1$*
    - ↳ *because we already start from good values with $v_{\pi^1}^0 = v_{\pi^0}^{+\infty}$*

With value iteration, we had to only evaluate optimal policy

- *say it takes $K_\star$ iterations: there is no reason that $K_\star$ be same as K_1 or K_2*
  - ↳ *each evaluation has a different initial and converging point*

↳ *in practice, it often happens that $K_\star > K_1$ and $K_\star \gg K_2$*
  - ↳ *so it might be that $K_\star \approx K_1 + K_2$*
  - ↳ *but usually withmultiple policy improvements, we see $K_\star < K_1 + K_2 + \dots$*

Complexity of Policy and Value Iteration

+ *If so, why should we use policy iteration?!*
– Well, not all problems are like a dummy grid world

In practice, it might be computationally hard to get very close to optimal values

- *in this case, we take non-converged values*
  ↳ *we consider them estimates of optimal values*
- *in value iteration we approximate optimal policy with on these estimates*
  ↳ *this might be a loose estimate*

If we do the same approximative computation with policy iteration

- *we often end up with a better policy*

Moral of Story

While value iteration typically show faster convergence, policy iteration can give better policies after convergence

Generalized Policy Iteration

In practice, we can terminate or change the order of computation in policy iteration to reduce its complexity: for instance, we could have

`GenPolicyItr():`

*Initiate with random $v_\pi(s)$ for all **non-terminal** states s*
*Set $v_\pi(s) = 0$ for **terminal** states s*
Initiate two random policies π and $\bar{\pi}$
while** $\pi \neq \bar{\pi}$ **do
    $v_\pi = \mathtt{TerminPolicyEval}(\pi, v_\pi)$ *and* $\pi \leftarrow \bar{\pi}$ changed
    $\bar{\pi} = \mathtt{PolicyImprov}(v_\pi)$
end while

where $\mathtt{TerminPolicyEval}(\pi, v_\pi)$ *evaluates policy π from starting value function v_π with a terminating recursion loop*

Generalized Policy Iteration: *Terminating Evaluation*

`TerminPolicyEval` (π, v_π^0):

Initiate values with v_π^0 *and set* $k = 0$
Make sure that $v_\pi^0(s) = 0$ *for* ***terminal*** *states* s
Choose a small threshold ϵ *and initiate* $\Delta = +\infty$ `# stopping criteria`
for $n = 1 : N$ **do**
  Compute $\bar{\mathcal{R}}_\pi(s^n) = \mathbb{E}_\pi\left\{\bar{\mathcal{R}}(s^n, a)\right\}$ `# average response`
end for
while $\Delta > \epsilon$ *and* $k < K$ **do** changed
  for $n = 1 : N$ **do**
    Update $v_\pi^{k+1}(s^n) = \bar{\mathcal{R}}_\pi(s^n) + \gamma \mathbb{E}_\pi\left\{v_\pi^k(\bar{S}) \,|s^n\right\}$ `# DP update`
  end for
  $\Delta = \max_n |v_\pi^{k+1}(s^n) - v_\pi^k(s^n)|$ `# check convergence`
  Update $k \leftarrow k + 1$
end while

Obviously, `TerminPolicyEval` (π, v_π) *does not return the exact values of the policy* π*, but only an estimate of them*

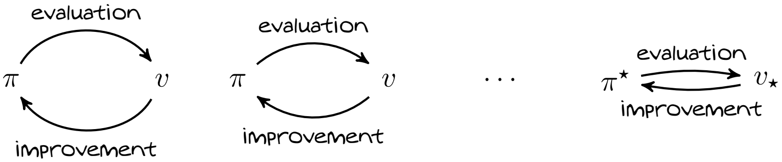
Generalized Policy Iteration

We can come up with various such ideas: *these variants are often called*

$$\textit{Generalized Policy Iteration} \equiv \textit{GPI}$$

These approaches all rely on

back-and-forth computation of policies and values



If designed properly, they all converge to optimal policy and optimal values

Some Final Remarks

+ *We know the algorithms now, but how can we guarantee that they converge? You showed us an simple example that recursion could simply diverge!*

– Well, we can show that what we discussed in this chapter converge: *it comes from the nice properties of Bellman equations*
  - ↳ *There are several proofs; for instance see a proof in Tom Mitchell's notes*

When it comes to practice, most known algorithms are proved to converge to optimal policy and optimal values; however, note that

- *Convergence guarantee is different from the speed of convergence*
  - ↳ *An algorithm might converge, but very slow*
- *If you deal with an unknown algorithm; then, you should make sure that it converges to optimal policy and optimal values*