Introduction to Reinforcement Learning

Part 1: MDP and Bellman Equations

Eric J. Pauwels

CWI & VU

Version, November 29, 2023

Outline

What is Reinforcement Learning (RL)

- RL is one of **three main learning problems** studied in AI:
- Learning by trial and error and improving over time;
- Natural form of learning, ubiquitous in biology and psychology:


What is Reinforcement Learning?

- **Supervised learning** needs teacher (labeled data!)
- **RL learns on the job:** interact, explore, exploit experience!

What makes RL attractive and interesting?

Natural way to learn complex skills, can we copy this?

RL Example 1: Playing a song by ear

- **Actions and transitions:** Moving from key to key...
- **Immediate reward:** correct key or not;
- **Cumulative reward:** Being able to play whole song
- Relatively easy, one gets immediate valuable feedback...

RL Example 2: Cooking

- **States:**
- **Actions:** e.g. cooking, seasoning, stirring, etc.
- **Transitions:** e.g. from raw to cooked
- **Immediate reward** e.g. smell, look
- **Final reward:** enjoyable dinner
- **Problem:** what's the best sequence of actions (i.e. optimal policy)?

RL Examples: Playing Chess

- **States:** board configurations
- **Actions:** legal moves
- **Value of state/action:** very valuable to determine next move (action),
- **Final reward:** Win(2), lose(0), draw(1)
- **Problem:** what's the optimal action to take in each state? (policy)

What makes Reinforcement Learning challenging?

1. **No Oracle**
    - Supervised learning: compare estimate to correct result (provided by oracle);
    - RL: no information on optimal action, just indicative *reward*
        - i.e. evaluative feedback, not prescriptive!
2. **Sparsity of feedback**
    - RL: only data on states/actions that have been experienced: (might be very small subset of entire state space);
    - For many RL problems, no feedback until target is reached!
    - Initial stages: just random search!
3. **Data generated during training**
    - Online learning: improve policies while interacting w. environment
    - No independent data generation!

policy $\longleftrightarrow$ data generation

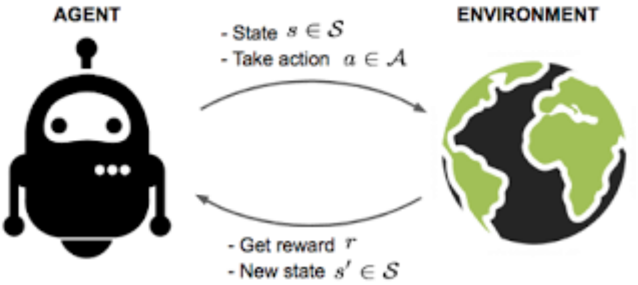
Reinforcement Learning: more abstract view

- **RL:** Learning from (sequential) interaction:
  take actions to move from state to (better!?) state;
- No labeled data, but **feedback** (reward, possibly delayed)
- **Goal:** optimise **end-result** (i.e. long-term/cumulative reward)

RL: Even more abstract version

Agent interacting with Environment:

- states(s), actions (a) and transitions: $s \xrightarrow{a} s'$ $(p(s' \mid s, a))$
- (immediate) reward $r(s, a, s')$

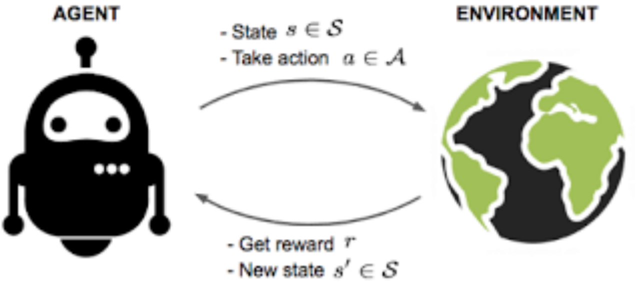


- **Example:** Taking a qualifying test (a), to pass from unqualified (s) to qualified state (s'), or not!

RL GOAL: Find optimal policy!

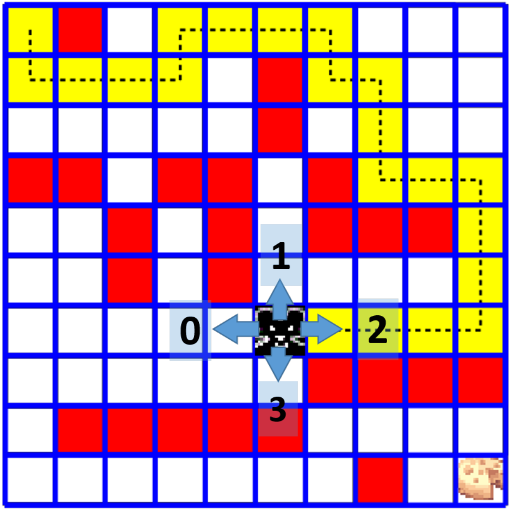
Policy π, tells for each state (s), which action (a) to take:

$$s \xrightarrow{\pi} a$$



RL Goal: Given environment, determine **optimal policy** π^*:
what **action** *to choose in* **each state** *to achieve* **best FINAL reward***?*

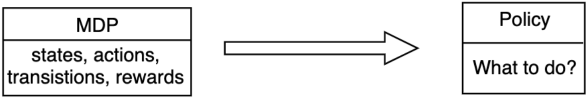
Gridworld Maze as Prototypical RL Example



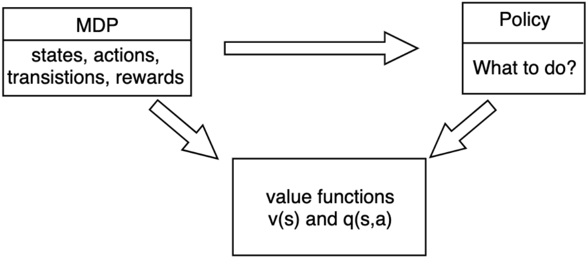
Overview

MDP
states, actions, transitions, rewards

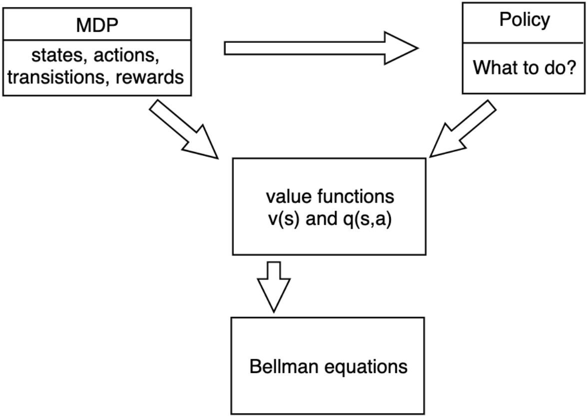
Overview



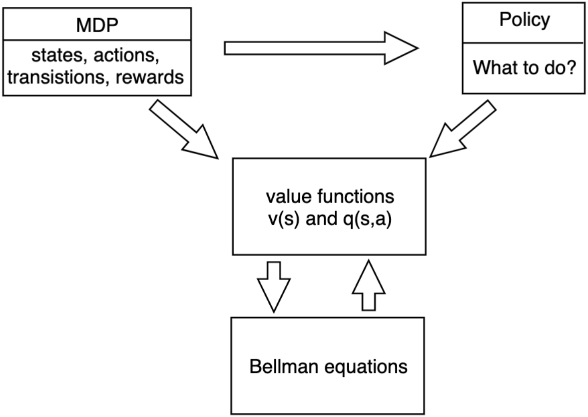
Overview


MDP
states, actions, transistions, rewards
Policy
What to do?
value functions
v(s) and q(s,a)

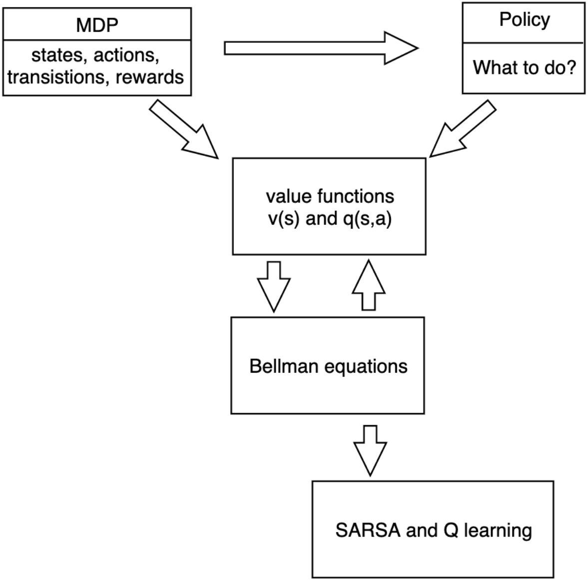
Overview


MDP
states, actions,
transistions, rewards
Policy
What to do?
value functions
v(s) and q(s,a)
Bellman equations

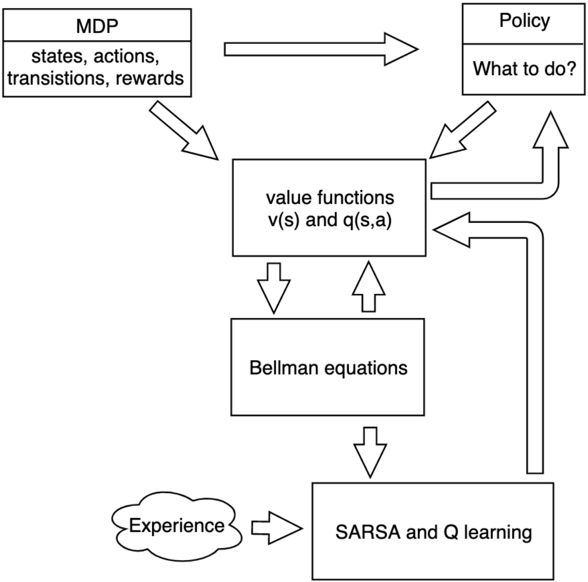
Overview


MDP
states, actions, transistions, rewards
Policy
What to do?
value functions v(s) and q(s,a)
Bellman equations

Overview


MDP
states, actions,
transistions, rewards
Policy
What to do?
value functions
v(s) and q(s,a)
Bellman equations
SARSA and Q learning

Overview


MDP
states, actions,
transistions, rewards
Policy
What to do?
value functions
v(s) and q(s,a)
Bellman equations
Experience
SARSA and Q learning

Outline

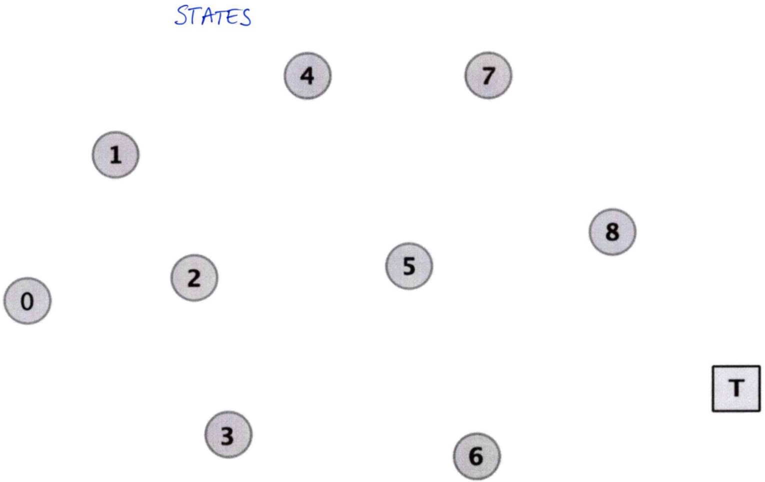
Markov Decision Process (MDP)

Mathematical formalism to capture the **essential characteristics** of the RL problem:

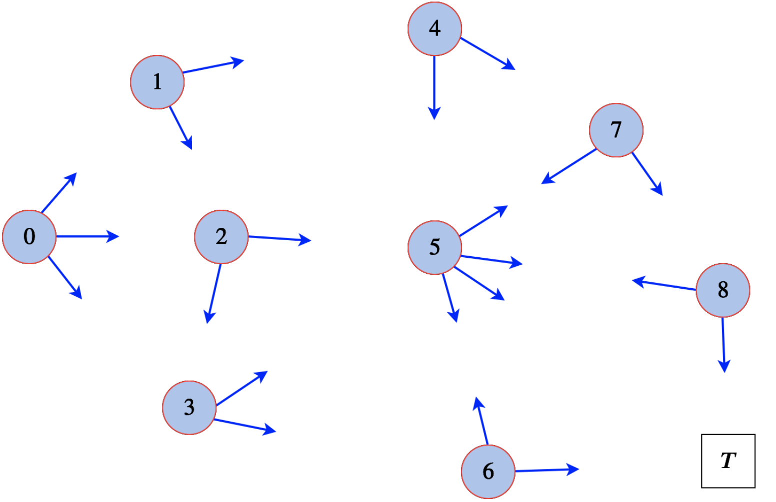
- MDP $=$ states, actions, transitions, rewards, (discount factor)

Markov Decision Process (MDP)
States

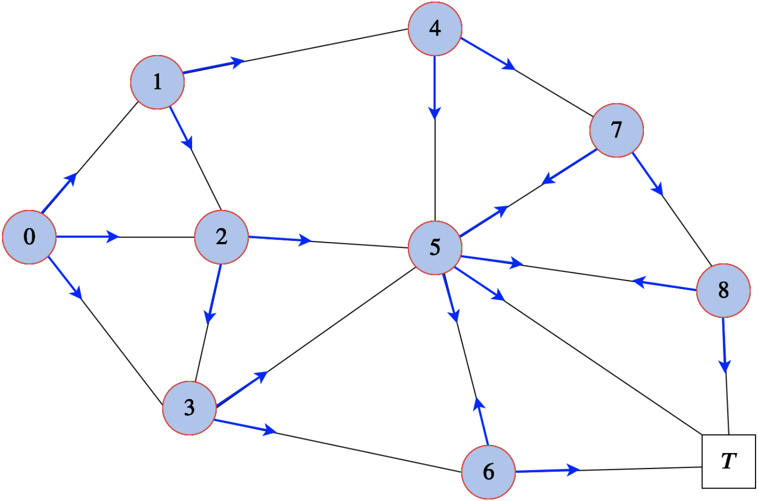
States capture the information that is available to agent



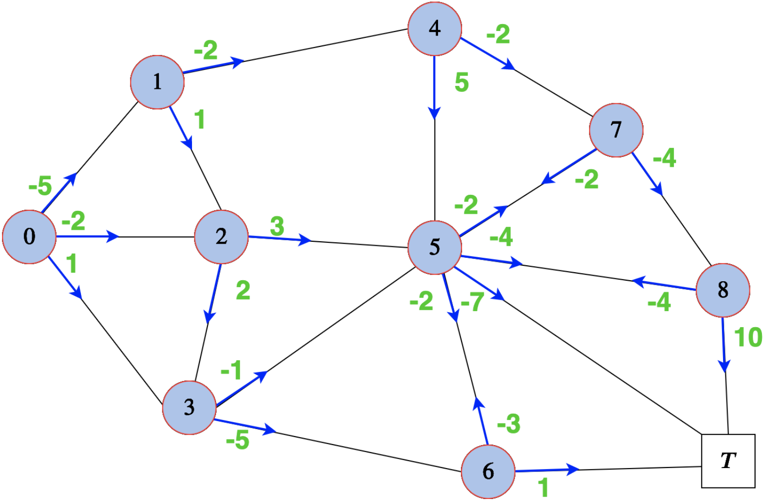
Markov Decision Process (MDP)
States + Actions



Markov Decision Process (MDP)
States + Actions + Transitions



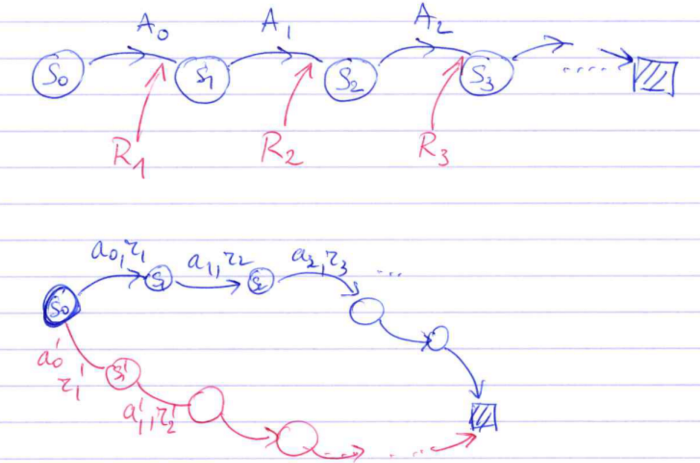
Markov Decision Process (MDP)
States + Actions + Transitions + Rewards



Markov decision processes (MDP)

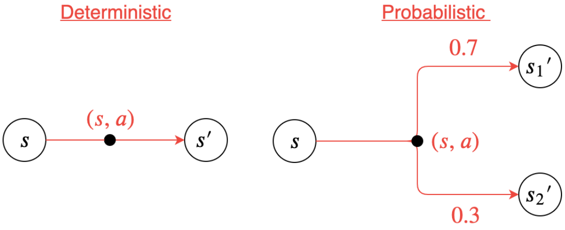
- **Markov decision processes (MDP)** provide a **formal model** for a **sequential decision problem**;
- A **finite MDP** $(\mathcal{S}, \mathcal{A}, \mathcal{P}, \mathcal{R}, \gamma)$ consists of:
  - **Discrete time** $t = 0, 1, 2, \ldots$
  - A discrete set of **states** $s \in S$ (captures relevant information)
  - A discrete set of **actions** $a \in A(s)$ for each s
  - A **transition function** $p(s'|s, a)$: probability of transitioning to state s' when taking action a at state s
    - e.g. prob of passing exam ($s \xrightarrow{a} s'$) when studying (a);
  - A **reward function** $r(s, a, s') = E[R \,|\, s, a, s']$: expected reward when taking action a at state s and transitioning to s'
    - Reward might depend on new state s' (e.g. $s' = pass$ or $s' = fail$)
  - A planning horizon H or **discount factor** γ;
    - How important are future rewards?
    - shortsighted $0 \leftarrow \gamma \rightarrow 1$ farsighted

Stochastic actions and returns: interpretation



Long-term return: $G_0(s_0) = R_1 + R_2 + \ldots + R_T = \sum_{t=1}^{T} R_t$

Detailed Definition and Notation (1)



$p(s' \mid s, a) = 1$

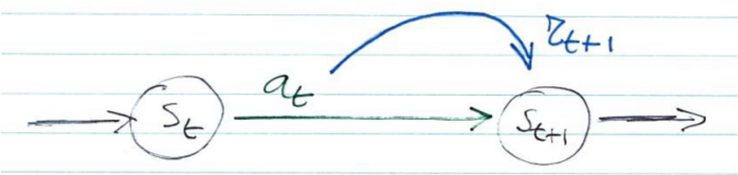
$p(s_1' \mid s, a) = 0.7$

$p(s_2' \mid s, a) = 0.3$

Transition probability to state s' when taking action a in state s :

$$p(s' \mid s, a) := P(S_{t+1} = s' \mid S_t = s, A_t = a)$$

Deterministic versus probabilistic transitions



- **Expected immediate reward** when transitioning from state s to state s' under action a:

$$r(s, a, s') := E(R_{t+1} = r \mid S_t = s, A_t = a, S_{t+1} = s')$$

The Markov property

Markov property: informally

- The present (state) has all the information necessary to predict the future: no need to keep track of the past.
- *"The future is independent of the past, given the present"*

$$P(S_{t+1}, R_{t+1} | s_t, a_t, r_t, s_{t-1}, a_{t-1}, \dots, r_1, s_0, a_0) = P(S_{t+1}, R_{t+1} | s_t, a_t)$$

If reward R_{t+1} does NOT depend on successor state S_{t+1}:

$$P(S_{t+1} | s_t, a_t, r_t, s_{t-1}, a_{t-1}, \dots, r_1, s_0, a_0) = P(S_{t+1} | s_t, a_t)$$

$$P(R_{t+1} | s_t, a_t, r_t, s_{t-1}, a_{t-1}, \dots, r_1, s_0, a_0) = P(R_{t+1} | s_t, a_t)$$

Joint versus marginal distribution

$$P(R_{t+1}, S_{t+1} \mid \ldots) = P(R_{t+1} \mid S_{t+1} \ldots) P(S_{t+1} \mid \ldots)$$

Dependence versus independence

- Action = hard work
- S_{t+1} = success or fail
- Reward R_{t+1}, could be:
  - dependent on next state: e.g. bonus if you succeed;
  - independent of next state, e.g. energy expenditure for action

Markov Property

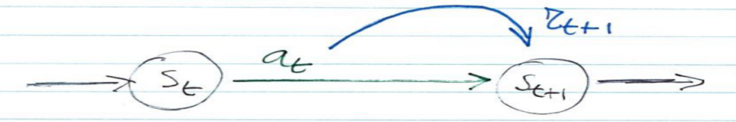
- MP simplifies the mathematics and makes it tractable;
- MP is sufficiently powerful as states can be expanded to include all the information necessary to predict future;
  - E.g.: to estimate the speed of a vehicle, we need at least two positions!

Example of Markov game

- **Game of chess**
- The current state of the board provides all information needed to determine next (optimal) move;
- No need to know the history (i.e. the path leading up to the current state)

Outline

Policies for a MDP

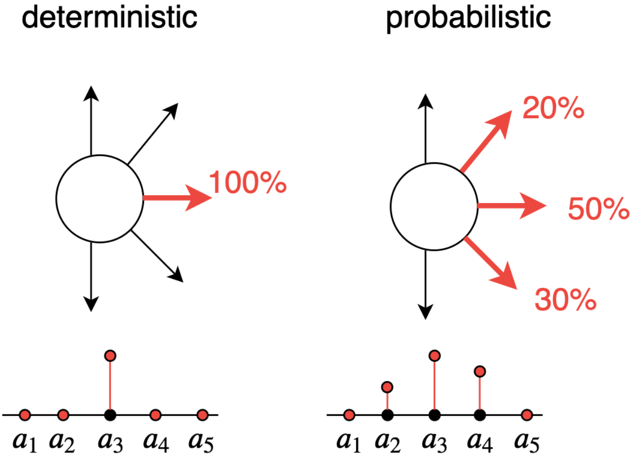


- **Policy** specifies what actions should be taken in a given state
- Hence, a **policy** π **maps states into actions;**
  - **Deterministic policy:** $a = \pi(s)$
  - **Stochastic policy:**

$$\pi(a \mid s) := P(A_{t+1} = a \mid S_t = s)$$

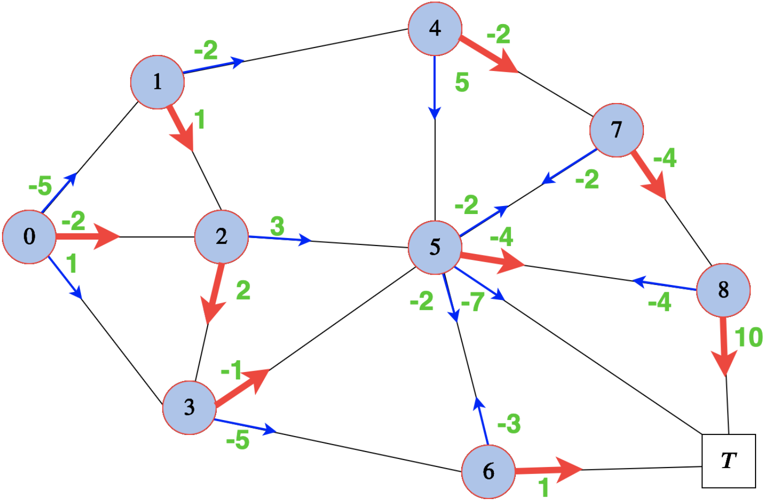
- **Important: policies** are **not** part of the MDP!

Deterministic vs. Probabilistic Policy

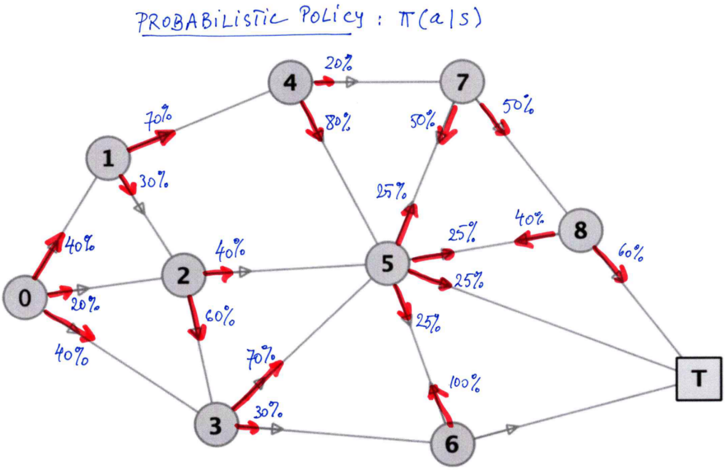


$$\boxed{s \to \pi(a \mid s)}$$

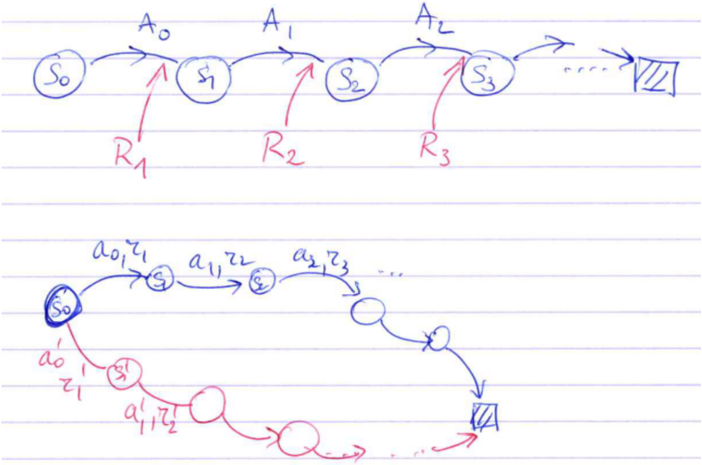
MDP + Policy (deterministic)



MDP + Policy (Probabilistic/Stochastic)



Stochastic actions and returns: interpretation



Long-term return: $G_0(s_0) = R_1 + R_2 + \ldots + R_T = \sum_{t=1}^{T} R_t$

Return and Rewards

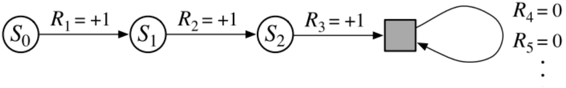
- The goal of the agent is to maximize the expected **(long-term) return**, a (discounted) sum over the immediate rewards received:

$$G_0 = R_1 + \gamma R_2 + \gamma^2 R_3 + \ldots = \sum_{k=1}^{\infty} \gamma^{k-1} R_k$$

or more generally:

$$G_t = R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+3} + \ldots = \sum_{k=1}^{\infty} \gamma^{k-1} R_{t+k}$$

- **Episodic tasks** can be interpreted as infinite-horizon, if we represent episode termination as transition to an **absorbing state** with self-transitions and zero reward.



Long-term (cumulative) return: some remarks

$$G_t = R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+3} + \ldots = \sum_{k=1}^{\infty} \gamma^{k-1} R_{t+k}$$

Remarks

- Notice that G depends on both the starting state (s) and the policy applied (π):

$$G \equiv G_\pi(s)$$

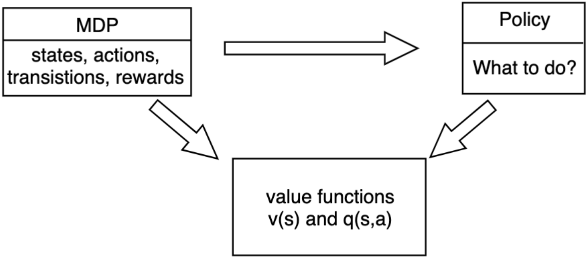
- Recursion relation:

$$\begin{aligned} G_t &= R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+3} + \ldots \\ &= R_{t+1} + \gamma(R_{t+2} + \gamma R_{t+3} + \ldots) \\ &= R_{t+1} + \gamma G_{t+1} \end{aligned}$$

Long-term reward

- Pole balancing

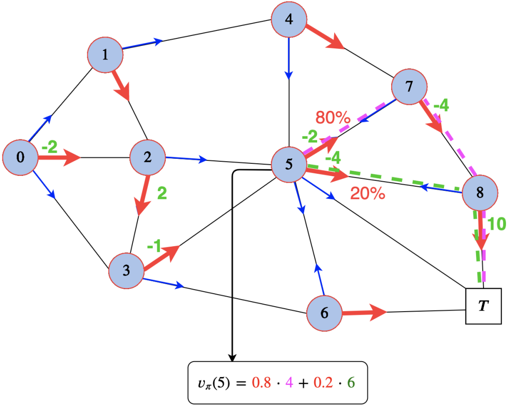
Overview


MDP
states, actions,
transistions, rewards
Policy
What to do?
value functions
v(s) and q(s,a)

MDP + Policy $\longrightarrow$ state value function $v_\pi(s)$

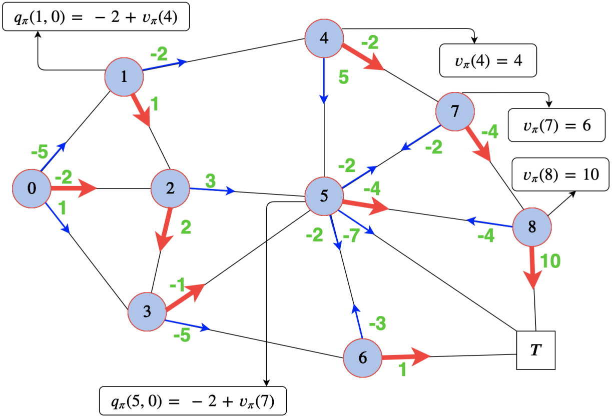
$v_\pi(s)$: expected value of state s when actions are specified by π:

$$v_\pi(s) = E_\pi\left(\sum_{t=1}^{T} R_t \mid s_0 = s\right)$$



MDP + Policy $\longrightarrow$ state-action value function $q_\pi(s, a)$

$$q_\pi(s, a) = E_\pi\left(\sum_{t=1}^{T} R_t \mid s_0 = s,\ a_0 = a\right)$$



Value functions: Tools for reasoning about future reward

- The **state value function** $v_\pi(s)$ assigns to each state s the **expected total return** when **starting** in that state s and **applying policy** π;

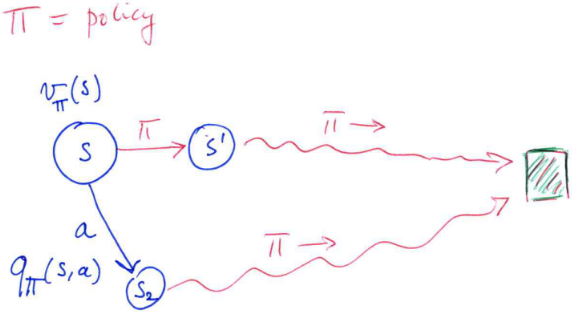
$$v_\pi(s) := E_\pi\Big[G_0 \,|\, S_0 = s\Big] = E_\pi\Big[\sum_{k=0}^{\infty} \gamma^k R_{k+1} | S_0 = s\Big]$$

- The **state-action value function** $q_\pi(s, a)$ of a policy π is:

$$q_\pi(s, a) := E_\pi\Big[G_0 \,|\, S_0 = s, A_0 = a\Big] = E_\pi\Big[\sum_{k=0}^{\infty} \gamma^k R_{k+1} \,|\, S_0 = s, A_0 = a\Big]$$

- $q(s, a)$ state space typically much larger than $v(s)$-state space! Hence, more difficult to learn.

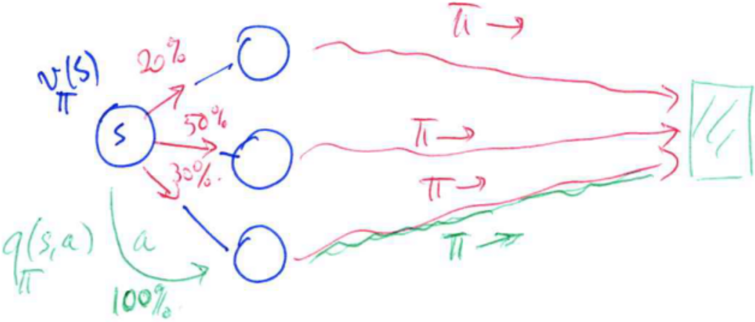
Difference between value functions $v_\pi(s)$ and $q_\pi(s, a)$



- $v_\pi(s)$: policy π dictates every action along the path;
- $q_\pi(s, a)$: first action a is taken independently of the policy π, from then onward, π dictates the remaining actions taken along the rest of the path.

Difference between value functions $v_\pi(s)$ and $q_\pi(s, a)$

Similar interpretation for stochastic policy.



Relationship between value and value-action function

Notice that $v(s)$ is a weighted mean of $q(s, a)$, where the weights are determined by the policy π:

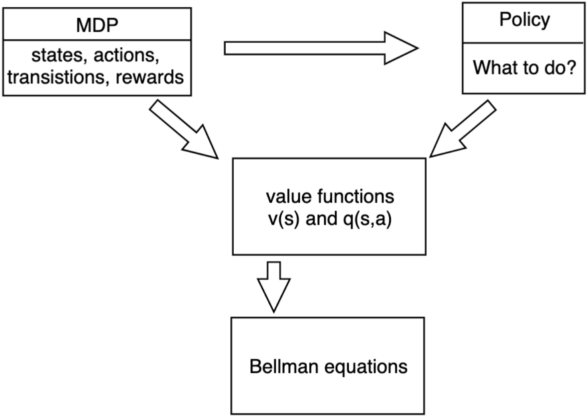
$$v_\pi(s) = \sum_a \pi(a \mid s) q_\pi(s, a)$$

Indeed:

$$\begin{aligned} v_\pi(s) &= E_\pi\Big[G_0 \mid S_0 = s\Big] \\ &= \sum_a E_\pi\Big[G_0 \mid S_0 = s, A_0 = a\Big]\pi(a \mid s) \\ &= \sum_a q_\pi(s, a)\pi(a \mid s) \end{aligned}$$

Outline

Overview

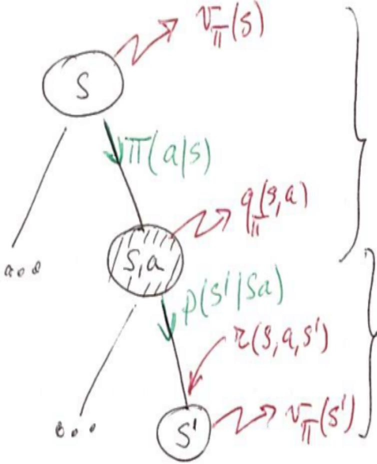

MDP
states, actions,
transistions, rewards
Policy
What to do?
value functions
v(s) and q(s,a)
Bellman equations

Bellman equation for value functions

Value function $v_\pi(s)$**:** $v_\pi(s) = \sum_a \pi(a \mid s) q_\pi(s,a)$
Value function $q_\pi(s,a)$

$$
\begin{aligned}
q_\pi(s,a) &= E_\pi\Big[G_0 \mid S_0 = s, A_0 = a\Big] \\
&= \sum_{s'} E_\pi\Big[G_0 \mid S_0 = s, A_0 = a, S_1 = s'\Big] p(s' \mid s,a) \\
&= \sum_{s'} E_\pi\Big[R_1 + \gamma G_1 \mid S_0 = s, A_0 = a, S_1 = s'\Big] p(s' \mid s,a) \\
&= \sum_{s'} \Big[r(s,a,s') + \gamma E_\pi(G_1 \mid S_1 = s')\Big] p(s' \mid s,a) \\
&= \sum_{s'} \Big[r(s,a,s') + \gamma E_\pi G_0(s')\Big] p(s' \mid s,a) \\
&= \sum_{s'} p(s' \mid s,a) \Big[r(s,a,s') + \gamma v_{\pi(}s')\Big]
\end{aligned}
$$

Bellman equations (schematically): Backup Diagram



$$\to v_\pi(s) = \sum_a \pi(a|s)\, q(s,a)$$

$$\to q(s,a) = \sum_{s'} p(s'|sa)\left[r(s,a,s') + \gamma V(s')\right]$$

Bellman equation: Summary

- **Back-up**

$$v_\pi(s) = \sum_a \pi(a \mid s) q(s,a)$$

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

- **Combined into recursion equations**

$$v_\pi(s) = \sum_a \pi(a \mid s) \sum_{s'} p(s' \mid s,a) \Big[r(s,a,s') + \gamma v_\pi(s') \Big]$$

$$q_\pi(s,a) = \sum_{s'} p(s' \mid s,a) \Big[r(s,a,s') + \gamma \sum_{a'} \pi(a' \mid s') q_\pi(s',a') \Big]$$

Bellman equation: Summary

- The definition of v_π can be rewritten recursively by making use of the transition model, yielding the **Bellman equation**:

$$v_\pi(s) = \sum_a \pi(a \mid s) \sum_{s'} p(s' \mid s, a)\Big[r(s, a, s') + \gamma v_\pi(s')\Big]$$

- This is a set of **linear equations,** one for each state, the solution of which defines the value of π
- A similar recursive relation holds for Q-values:

$$q_\pi(s, a) = \sum_{s'} p(s' \mid s, a)\Big[r(s, a, s') + \gamma \sum_{a'} \pi(a' \mid s') q_\pi(s', a')\Big]$$

Matrix form of Bellman equation

$$v_\pi(s) = \sum_a \pi(a \mid s) \sum_{s'} p(s' \mid s, a) \Big[r(s, a, s') + \gamma v_\pi(s') \Big]$$

We can rewrite this as:

$$\begin{aligned} v_\pi(s) \quad &= \quad \gamma \sum_{s'} \underbrace{\left(\sum_a \pi(a \mid s) p(s' \mid s, a) \right)}_{P_\pi(s,s')} v_\pi(s') \\ &\quad + \sum_a \underbrace{\left(\sum_{s'} p(s' \mid s, a) r(s, a, s') \right)}_{R(s,a)} \pi(a \mid s) \end{aligned}$$

Matrix form of Bellman equation

Under policy π:

- square matrix $P_\pi(s, s')$ is **transition probability** $s \to s'$:

$$P_\pi(s, s') := \sum_a \pi(a \mid s) p(s' \mid s, a)$$

- $R(s, a)$ is the **expected (immediate) reward** when taking action a in state s:

$$R(s, a) = \sum_{s'} p(s' \mid s, a) r(s, a, s')$$

- $\mathbf{r}_\pi(s)$ is the **expected (immediate) reward** in state s:

$$\mathbf{r}_\pi(s) = \sum_a \pi(a \mid s) R(s, a)$$

Matrix formulation of Bellman equation

- Bellman in matrix form:

$$\mathbf{v}_\pi = \gamma P_\pi \mathbf{v}_\pi + \mathbf{r}_\pi \qquad \text{or again} \qquad (I - \gamma P_\pi)\mathbf{v}_\pi = \mathbf{r}_\pi$$

- Solving for the value function v_π:

$$\mathbf{v}_\pi = (I - \gamma P_\pi)^{-1}\mathbf{r}_\pi$$

- Notice that:

$$\mathbf{v} = (I - \gamma P)^{-1}\mathbf{r} = (I + \gamma P + \gamma^2 P^2 + \ldots)\mathbf{r}$$

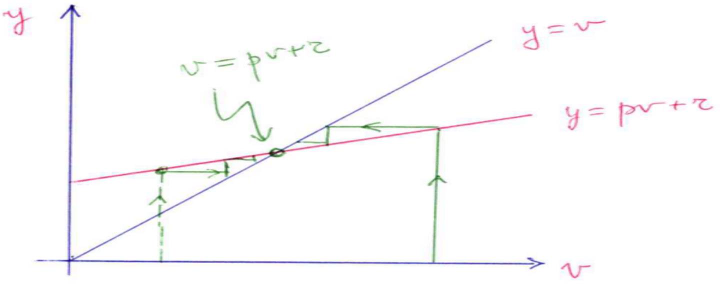
Solving Bellman eqs: Iteration to Fix-Point

Matrix form of **Bellman equation** corresponds to fix-point:

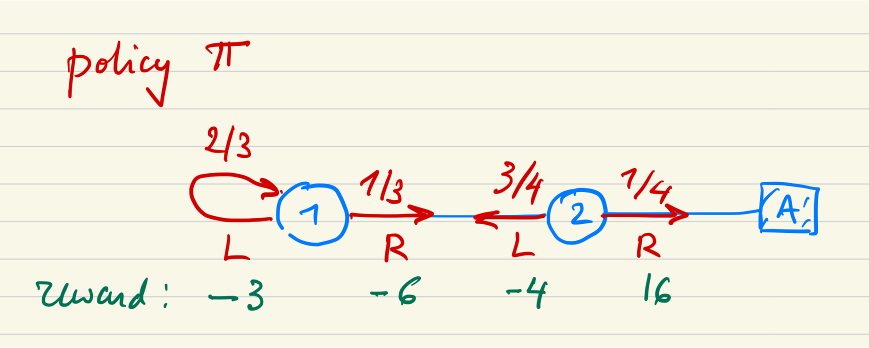
$$\mathbf{v}_\pi = \gamma P_\pi \mathbf{v}_\pi + \mathbf{r}_\pi \qquad \Longrightarrow \qquad \mathbf{v}_\pi = (I - \gamma P_\pi)^{-1} \mathbf{r}_\pi$$

Iterative solution (fix-point solution): update rule:

$$\mathbf{v}^{k+1} = \gamma P \mathbf{v}^k + \mathbf{r}$$

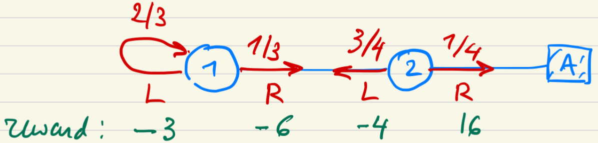


Solving Bellman eqs: Worked example



Solving Bellman eqs: Worked example

policy π



$$P_\pi = \begin{array}{c|ccc} s \backslash s' & A & 1 & 2 \\ \hline A & 1 & 0 & 0 \\ 1 & 0 & 2/3 & 1/3 \\ 2 & 1/4 & 3/4 & 0 \end{array}$$

$$R(s,a) = \begin{array}{c|cc} & L & R \\ \hline A & 0 & 0 \\ 1 & -3 & -6 \\ 2 & -4 & 16 \end{array}$$

Solving Bellman eqs: Worked example

$$R(s,a) = \begin{array}{c|cc} & L & R \\ \hline A & 0 & 0 \\ 1 & -3 & -6 \\ 2 & -4 & 16 \end{array}$$

$$\boxed{r_\pi(s) = \sum_a R(s,a)\,\pi(a|s)}$$

$$r(1) = -3 \cdot \frac{2}{3} + (-6)\frac{1}{3} = -4$$

$$r(2) = -4 \cdot \frac{3}{4} + 16 \cdot \frac{1}{4} = 1$$

$$r(A) = 0$$

$$r_\pi = (0, -4, 1)^T$$

Solving Bellman eqs: Worked example

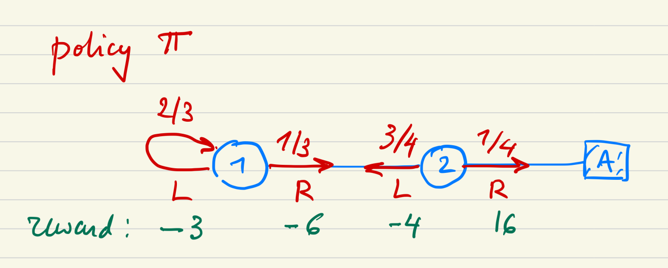
$$\boxed{v_\pi = \gamma P_\pi v_\pi + r_\pi} \qquad \gamma = 1$$

$$\underbrace{\begin{pmatrix} v(A) \\ v(1) \\ v(2) \end{pmatrix}}_{v} = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 2/3 & 1/3 \\ 1/4 & 3/4 & 0 \end{pmatrix} v + \begin{pmatrix} 0 \\ -4 \\ 1 \end{pmatrix}$$

$$v_0 = r \;, \quad v_1 = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 2/3 & 1/3 \\ 1/4 & 3/4 & 0 \end{pmatrix} \begin{pmatrix} 0 \\ -4 \\ 1 \end{pmatrix} + \begin{pmatrix} 0 \\ -4 \\ 1 \end{pmatrix}$$

$$= \begin{pmatrix} 0 \\ -7/3 \\ -3 \end{pmatrix} + \begin{pmatrix} 0 \\ -4 \\ 1 \end{pmatrix} = \begin{pmatrix} 0 \\ -19/3 \\ -2 \end{pmatrix}$$

Solving Bellman eqs: Worked example



$$
\begin{aligned}
v_2 &= (0, -6.3333, -2.0000) \\
v_3 &= (0, -8.8889, -3.7500) \\
\ldots. &= \ldots. \\
v_\infty &= (0, -44, -32)
\end{aligned}
$$

Example: MDP + value functions

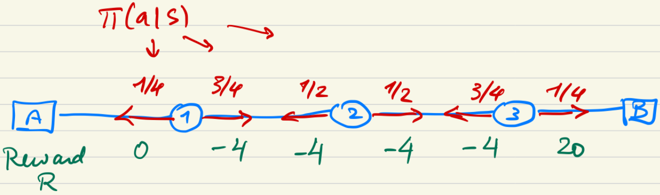
Example: - MDP + policy : see fig.

Assumptions:

1. No discounting: $\gamma = 1$
2. Deterministic transitions: $s \xrightarrow{a} s'$

   i.e.: $p(s' \mid s, a)$ = degenerate.

   ↳ either 1 or 0.



Example: MDP + value functions

$$P_\pi(s,s') = \sum_a \pi(a|s)\, p(s'|s,a)$$

$$= P(\text{transition } s \to s' \text{ in 1 time step})$$

$$P = \begin{array}{c|ccccc} s \backslash s' & A & 1 & 2 & 3 & B \\ \hline A & 1 & 0 & 0 & 0 & 0 \\ 1 & 1/4 & 0 & 3/4 & 0 & 0 \\ 2 & 0 & 1/2 & 0 & 1/2 & 0 \\ 3 & 0 & 0 & 3/4 & 0 & 1/4 \\ B & 0 & 0 & 0 & 0 & 1 \end{array}$$

Example: MDP + value functions

$$R(s,a) = \sum_{s'} p(s'|s,a)\, r(s,a,s')$$

= expected immediate reward when taking action a in state s.

$$R = \begin{array}{c} \\ A \\ 1 \\ 2 \\ 3 \\ B \end{array} \begin{array}{c} \begin{array}{cc} L & R \end{array} \\ \boxed{\begin{array}{cc} 0 & 0 \\ 0 & -4 \\ -4 & -4 \\ -4 & 20 \\ 0 & 0 \end{array}} \end{array}$$

Example: MDP + value functions

$$r(s) = \sum_a R(s,a)\,\pi(a|s)$$

$$r = \boxed{\begin{matrix} r(A) \\ r(1) \\ r(2) \\ r(3) \\ r(B) \end{matrix}} = \underline{\text{diag}} \left(\underbrace{\begin{array}{c} \begin{matrix} L & R \end{matrix} \\ \boxed{\begin{matrix} 0 & 0 \\ 0 & -4 \\ -4 & -4 \\ -4 & 20 \\ 0 & 0 \end{matrix}} \end{array}}_{R(s,a)} \; \underbrace{\begin{array}{cc} & \begin{matrix} A & 1 & 2 & 3 & B \end{matrix} \\ \begin{matrix} L \\ R \end{matrix} & \boxed{\begin{matrix} * & 1/4 & 1/2 & 3/4 & * \\ * & 3/4 & 1/2 & 1/4 & * \end{matrix}} \end{array}}_{\pi(a|s)} \right)$$

Example: MDP + value functions

$$r(s) = \sum_a R(s,a)\,\pi(a|s)$$

= expected immediate return in s

$$r = \begin{bmatrix} 0 & \left(\frac{1}{4}\cdot 0 + \frac{3}{4}(-4)\right) = -3 & -4 & \frac{3}{4}(-4) + \frac{1}{4}20 = 2 & 0 \end{bmatrix}$$

A 1 2 3 B

Example: MDP + value functions

Bellman eqs in matrix form:

$$v = \begin{bmatrix} v(A) \\ v(1) \\ v(2) \\ v(3) \\ v(B) \end{bmatrix} = \begin{array}{c} \begin{array}{ccccc} A & 1 & 2 & 3 & B \end{array} \\ \begin{bmatrix} 1 & 0 & 0 & 0 & 0 \\ 1/4 & 0 & 3/4 & 0 & 0 \\ 0 & 1/2 & 0 & 1/2 & 0 \\ 0 & 0 & 3/4 & 0 & 1/4 \\ 0 & 0 & 0 & 0 & 1 \end{bmatrix} \end{array} v + \underbrace{\begin{bmatrix} 0 \\ -3 \\ -4 \\ 2 \\ 0 \end{bmatrix}}_{=r}$$

$$v = P \cdot v + r$$

Example:

$$v(2) = \tfrac{1}{2} v(1) + \tfrac{1}{2} v(3) + (-4)$$

$$v(3) = \tfrac{3}{4} v(2) + \tfrac{1}{4} \underbrace{v(B)}_{=0} + 2$$

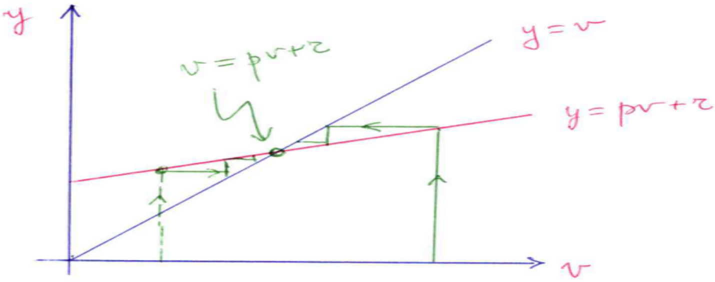
Solving Bellman eqs: Iteration to Fix-Point

Matrix form of **Bellman equation** corresponds to fix-point:

$$\mathbf{v}_\pi = \gamma P_\pi \mathbf{v}_\pi + \mathbf{r}_\pi \qquad \Longrightarrow \qquad \mathbf{v}_\pi = (I - \gamma P_\pi)^{-1} \mathbf{r}_\pi$$

Iterative solution (fix-point solution): update rule:

$$\mathbf{v}^{k+1} = \gamma P \mathbf{v}^k + \mathbf{r}$$

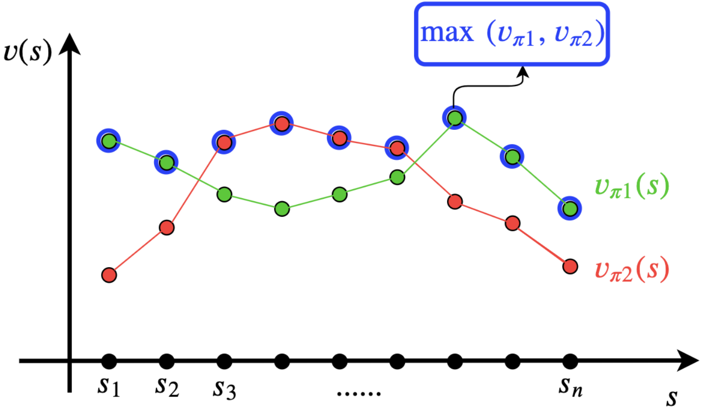


Outline

Optimal value functions

The optimal value functions are defined by the **pointwise maximum over policies**:

$$\forall s, a : \quad v^*(s) := \max_\pi v_\pi(s) \quad \text{and} \quad q^*(s,a) := \max_\pi q(s,a)$$

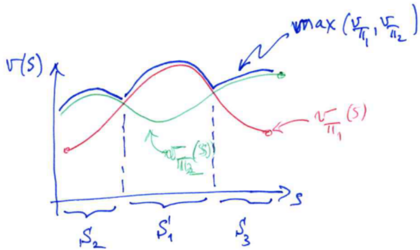


Optimal policy π^*

There exists an **optimal policy** π^* such that its value functions corresponds to the optimal value functions:

$$v_{\pi^*}(s) = v^*(s) \qquad \text{and} \qquad q_{\pi^*}(s,a) = q^*(s,a)$$

Intuition:



$$\pi^*(s) = \begin{cases} \pi_1(s) & \text{if } s \in S_2 \\ \pi_2(s) & \text{if } s \in S_1 \cup S_3 \end{cases}$$

Optimal value functions

- Value functions define a partial ordering over policies:

$$\pi \succ \pi' \Rightarrow v_{\pi(s)} \geq v_{\pi'}(s), \forall s \in S$$

- There can be multiple optimal policies but they all share the same **optimal state-value function**:

$$v^*(s) = \max_\pi v_\pi(s), \quad \forall s \in S$$

- They also share the same **optimal action-value function**:

$$q^*(s,a) = \max_\pi q_\pi(s,a), \quad \forall s \in S, a \in A$$

Optimal value functions: from weighted mean to max

- State-value function
  - General:

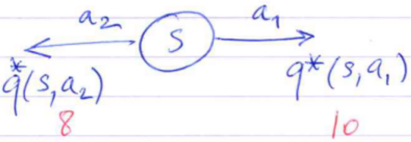
$$v_\pi(s) = \sum_a \pi(a \mid s) q_\pi(s, a)$$

  - Optimal

$$v^*(s) = v_{\pi^*}(s) = \max_a q_{\pi^*}(s, a) = \max_a q^*(s, a)$$

$$v^*(s) = \max_a q^*(s, a)$$

Bellman Optimality Equations



$q^*(s, \underline{a_1})$ = best possible expected return if taking action $\underline{\underline{a_1}}$ in s. $= 10$

$q^*(s, \underline{\underline{a_2}})$ = best possible expected return if taking action $\underline{\underline{a_2}}$ in s $= 8$

$v^*(s)$ = best possible expected return in s.

$$\boxed{v^*(s) = \max_a q^*(s,a)} \quad (= 10).$$

Bellman Optimality Equations

Deterministic transition $s \xrightarrow{a} s'$

$$\underbrace{q^*(s,a)}_{\downarrow} = r(s,a,s') + \gamma \underbrace{v^*(s')}_{\downarrow}$$

Best possible expected Return when Committed to action a

Best possible expected return in this state s'

$$q^*(s,a) = \sum_{s'} p(s'|sa)\left[r(s,a,s') + \gamma v^*(s')\right]$$

General

Optimal value functions: from weighted mean to max

- State-action value function
  - General:

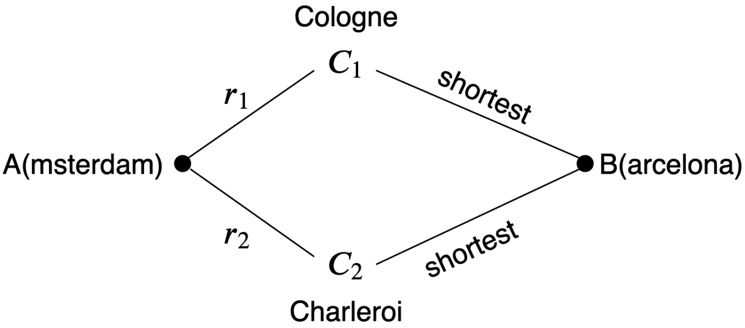
$$q_\pi(s,a) = \sum_{s'} p(s' \mid s,a)(r(s,a,s') + \gamma v_\pi(s))$$

  - Optimal:

$$q_{\pi^*}(s,a) = \sum_{s'} p(s' \mid s,a)(r(s,a,s') + \gamma v_{\pi^*}(s))$$

$$q^*(s,a) = \sum_{s'} p(s' \mid s,a)(r(s,a,s') + \gamma v^*(s))$$

Bellman Optimality equation: Travel Distance Analogy



$$d^*(A, B) = \min_{C_i} \{r_i + d^*(C_i, B)\}$$

Bellman optimality conditions (for deterministic transitions)

- Travel Analogy: Shortest distance paths:

$$d^*(A, B) = \min_{C_i} \{r_i + d^*(C_i, B)\}$$

- Deterministic transitions: $s \xrightarrow{a} s_a$

$$v^*(s) = \max_a \{r(s, a, s_a) + v^*(s_a)\}$$

$$q^*(s, a) = r(s, a, s_a) + v^*(s) = r(s, a, s_a) + \max_{a'} q^*(s_a, a')$$

Bellman optimality equations (General form)

- **Optimal state value function**

$$v^*(s) = \max_a q^*(s, a)$$

- **Optimal state-action value function**

$$q^*(s, a) = \sum_{s'} p(s' \mid s, a) \left[r(s, a, s') + \gamma v^*(s') \right]$$

- **Combined**

$$v^*(s) = \max_{a \in A} \sum_{s'} p(s' \mid s, a) \Big[r(s, a, s') + \gamma v^*(s') \Big]$$

$$q^*(s, a) = \sum_{s'} p(s' \mid s, a) \Big[r(s, a, s') + \gamma \max_{a' \in A} q^*(s', a') \Big]$$

Backup Diagram for Bellman Optimality Equations

Optimize over actions!

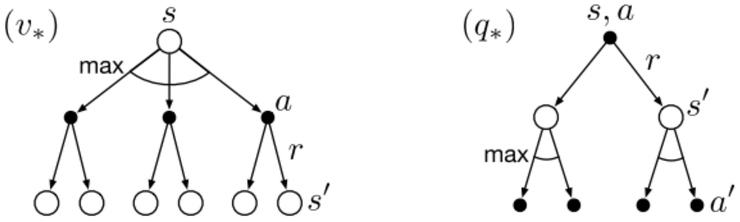


Figure 3.5: Backup diagrams for v_* and q_*

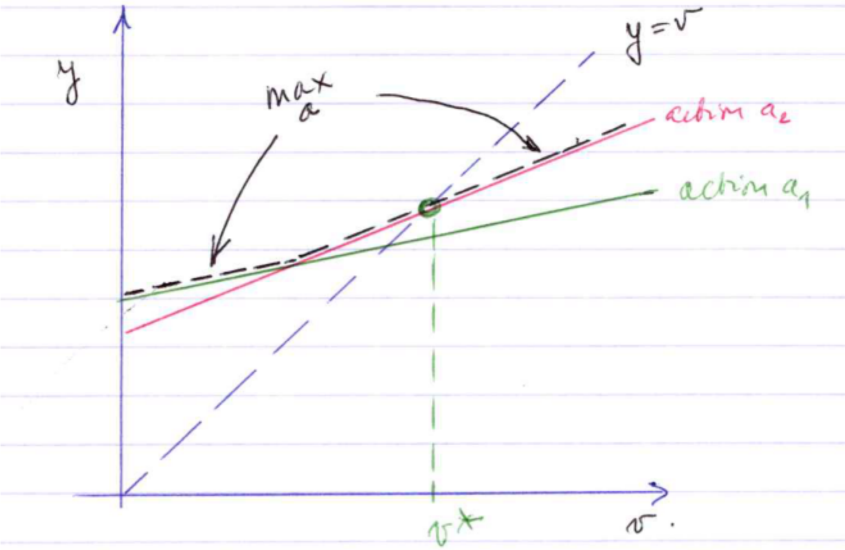
Bellman optimality equation in matrix form

$$
\begin{aligned}
v^*(s) &= \max_{a \in A} \sum_{s'} p(s' \mid s, a)\Big[r(s, a, s') + \gamma v^*(s')\Big] \\
&= \max_{a} \left(R(s, a) + \gamma \sum_{s'} \underbrace{p(s' \mid s, a)}_{T_a(s,s')} v^*(s') \right) \\
&= \max_{a} \left(R(s, a) + \gamma \sum_{s'} T_a(s, s') v^*(s') \right)
\end{aligned}
$$

or in matrix notation:

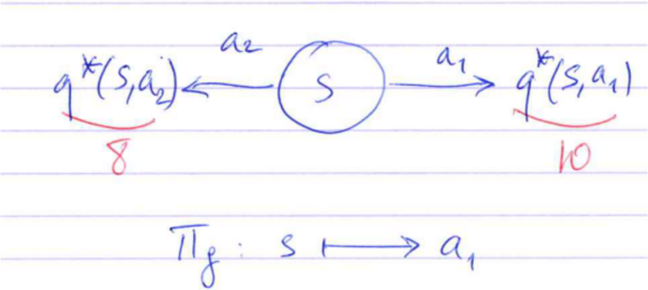
$$
\mathbf{v}^* = \max_{a} \left(R_a + \gamma T_a \mathbf{v}^* \right)
$$

Bellman optimality equation



Importance of v^* and q^*

- Markov property: Optimal decision only depends on current state:
- **Greedification:** In state s pick action a with highest $q^*(s, a)$



Bellman Optimality Conditions

- v^* and q^* satisfy a set of (non-linear) equations analogous to Bellman equations for v_π and q_π;
- these non-linear equations do not refer explicitly to the optimal policy;
- As a consequence we can find the optimal policy π^* by
  1. first solving these equations to find $v^*(s)$ (and $q^*(s,a)$) and
  2. then use **greedification** to determine the corresponding optimal policy π^*:

$$\forall s : \quad a_{opt} = \arg\max_a q^*(s, a)$$

Example: Bellman optimality conditions

MDP

Example : - MDP + policy : see fig.

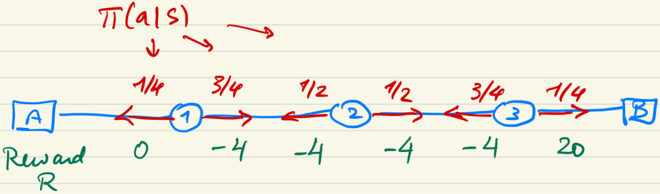
Assumptions :

① No discounting, $\gamma = 1$

② Deterministic transitions, $s \xrightarrow{a} s'$

i.e. : $\cancel{p}(s'|s,a)$ = degenerate.

↳ either 1 or 0.



Example: Bellman optimality conditions

Bellman Optimality Conditions

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

$$= \max_a \left\{ R(s,a) + \gamma \sum_{s'} p(s'|s a)\, v^*(s') \right\}$$

Figure: General expression for v^*

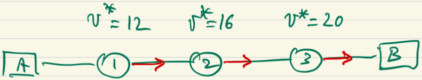
Bellman optimality conditions: Deterministic transition

Deterministic transition : $s \xrightarrow{a} s_a$

$$v^*(s) = \max_a \left[R(s,a) + \gamma v^*(s_a) \right]$$

In this example, optimal policy is simple

$\boxed{\pi^* : \text{always move right}}$



Eg, $S = 2$

$a = L$: $S_a = 1$, $v^*(S_a) = 12$, $R(s,a) = -4$

$R(s,a) + v^*(S_a) = -4 + 12 = 8$

$a = R$: $S_a = 3$, $v^*(S_a) = 20$, $R(s,a) = -4$

$R(s,a) + v^*(S_a) = -4 + 20 = \boxed{16}$

Bellman optimality conditions: Deterministic transition

Computing v^*, Worked example.

$$R = \begin{bmatrix} 0 & 0 \\ 0 & -4 \\ -4 & -4 \\ -4 & 20 \\ 0 & 0 \end{bmatrix}$$

Deterministic: $s \xrightarrow{a} s_a$

$$v^*(s) = \max_a \left[R(s,a) + \gamma v^*(s_a) \right]$$

$\gamma = 1$

Figure: $R(s, a)$ and $v^*(s)$ for deterministic transitions

Bellman optimality condition: Computing v^* using iteration

Initialise

$t=0 \longrightarrow t=1$

$$v^* = 0 \qquad v^* = \max_a \left(\begin{array}{c} \overbrace{\begin{array}{cc} L & R \end{array}}^{a} \\ \begin{bmatrix} 0 & 0 \\ 0 & -4 \\ -4 & -4 \\ -4 & 20 \\ 0 & 0 \end{bmatrix} \end{array} + \begin{bmatrix} 0 & 0 \\ 0 & 0 \\ 0 & 0 \\ 0 & 0 \\ 0 & 0 \end{bmatrix} \right) = \begin{pmatrix} 0 \\ 0 \\ -4 \\ 20 \\ 0 \end{pmatrix}$$

$$t=2: \quad v^* = \max_a \left\{ \begin{bmatrix} 0 & 0 \\ 0 & -4 \\ -4 & -4 \\ -4 & 20 \\ 0 & 0 \end{bmatrix} + \begin{bmatrix} 0 & 0 \\ 0 & -4 \\ 0 & 20 \\ -4 & 0 \\ 0 & 0 \end{bmatrix} \right\} = \begin{bmatrix} 0 \\ 0 \\ 16 \\ 20 \\ 0 \end{bmatrix}$$

Example: Bellman optimality conditions

$$t=3: \quad v^{*} = \max_{a} \left\{ \begin{bmatrix} 0 & 0 \\ 0 & -4 \\ -4 & -4 \\ -4 & 20 \\ 0 & 0 \end{bmatrix} + \begin{bmatrix} 0 & 0 \\ 0 & 16 \\ 0 & 20 \\ 16 & 0 \\ 0 & 0 \end{bmatrix} \right\} = \begin{bmatrix} 0 \\ 12 \\ 16 \\ 20 \\ 0 \end{bmatrix}$$

Figure: $t = 3$

Example: Bellman optimality conditions

$\underline{t=4}$

$$v^* = \max_a \left\{ \begin{bmatrix} 0 & 0 \\ 0 & -4 \\ -4 & -4 \\ -4 & 20 \\ 0 & 0 \end{bmatrix} + \begin{bmatrix} 0 & 0 \\ 0 & 16 \\ 12 & 20 \\ 16 & 0 \\ 0 & 0 \end{bmatrix} \right\} = \begin{bmatrix} 0 \\ 12 \\ 16 \\ 20 \\ 0 \end{bmatrix}$$

Converged!

Figure: Convergence for $t = 4$

For known MDP: compute $q^*(s, a)$ based on $v^*(s)$

$$s \xrightarrow{a} s_a$$

$$q^*(s,a) = r(s,a,s_a) + \gamma v^*(s_a)$$

$$v^* = \begin{array}{cc} \begin{pmatrix} 0 \\ 12 \\ 16 \\ 20 \\ 0 \end{pmatrix} & \begin{array}{c} A \\ 1 \\ 2 \\ 3 \\ B \end{array} \end{array} \qquad q^* = \begin{array}{cc} & \begin{array}{cc} L & R \end{array} \\ \begin{array}{c} A \\ 1 \\ 2 \\ 3 \\ B \end{array} & \underbrace{\begin{bmatrix} 0 & 0 \\ 0 & -4 \\ -4 & -4 \\ -4 & 20 \\ 0 & 0 \end{bmatrix}}_{R(s,a)} \end{array} + \underbrace{\begin{bmatrix} 0 & 0 \\ 0 & 16 \\ 12 & 20 \\ 16 & 0 \\ 0 & 0 \end{bmatrix}}_{v^*(s_a)} = \begin{bmatrix} 0 & 0 \\ 0 & 12 \\ 8 & 16 \\ 12 & 20 \\ 0 & 0 \end{bmatrix}$$

$$a^*(s) = \arg\max\ q^*(s,a)$$

Hence $a^*(s) = R$ for $s = 1, 2, 3$

Figure: Computing $q^*(s, a)$ from $v^*(s)$

Bellman optimality equations

If MDP is fully known, it suffices to compute v^*, as q^* can then be derived:

- **Optimal state-action value function**

$$q^*(s,a) = \sum_{s'} p(s' \mid s,a) \left[r(s,a,s') + \gamma v^*(s') \right]$$

- **Recall: Optimal state value function**

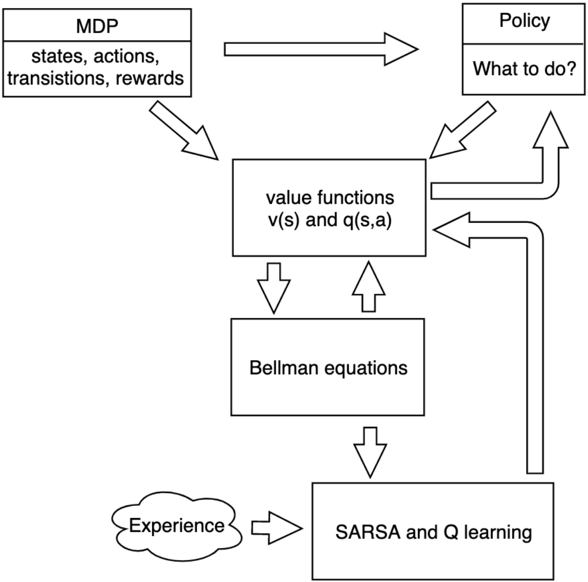
$$v^*(s) = \max_a q^*(s,a)$$

Model-based vs model-free

- **Model-based (PLANNING):** the MDP $= (\mathcal{S}, \mathcal{A}, \mathcal{P}, \mathcal{R}, \gamma)$ is completely specified;
  - Solve the Bellman equations (DP)!
- **Model-free (LEARNING):** only **direct experience**, i.e. only sample paths (states, actions and rewards) are given. Put differently, only experience-based information is given!
  - Random search but Bellman equations allow to propagate values!

Outline

Overview


MDP
states, actions, transistions, rewards
Policy
What to do?
value functions
v(s) and q(s,a)
Bellman equations
Experience
SARSA and Q learning

Bellman equations: General versus Optimal

- **State value functions**:
  - General:

$$v_\pi(s) = \sum_a \pi(a \mid s) \sum_{s'} p(s' \mid s, a) \Big[r(s, a, s') + \gamma v_\pi(s') \Big]$$

  - Optimal:

$$v^*(s) = \max_a \sum_{s'} p(s' \mid s, a) \Big[r(s, a, s') + \gamma v^*(s') \Big]$$

- **State-action value function**
  - General:

$$q_\pi(s, a) = \sum_{s'} p(s' \mid s, a) \Big[r(s, a, s') + \gamma \sum_{a'} \pi(a' \mid s') q_\pi(s', a') \Big]$$

  - Optimal:

$$q^*(s, a) = \sum_{s'} p(s' \mid s, a) \Big[r(s, a, s') + \gamma \max_{a'} q^*(s', a') \Big]$$

Taxonomy of RL problems

	Prediction *Estimation:* *Given π, what is v?*	**(Optimal) Control** *Optimisation:* *What is optimal π?*
model-based (MDP given)	Policy evaluation using Dyn. Programming (DP)	Policy improvement (+ Policy evaluation) = Policy iteration
model-free (MDP unknown)	Monte Carlo (MC) Temporal Diff^{ing} (TD) = "impatient MC" *bootstrapping!*	Generalized Policy Iteration *"simultaneous"*