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 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!

$$\text{policy} \longleftrightarrow \text{data generation}$$

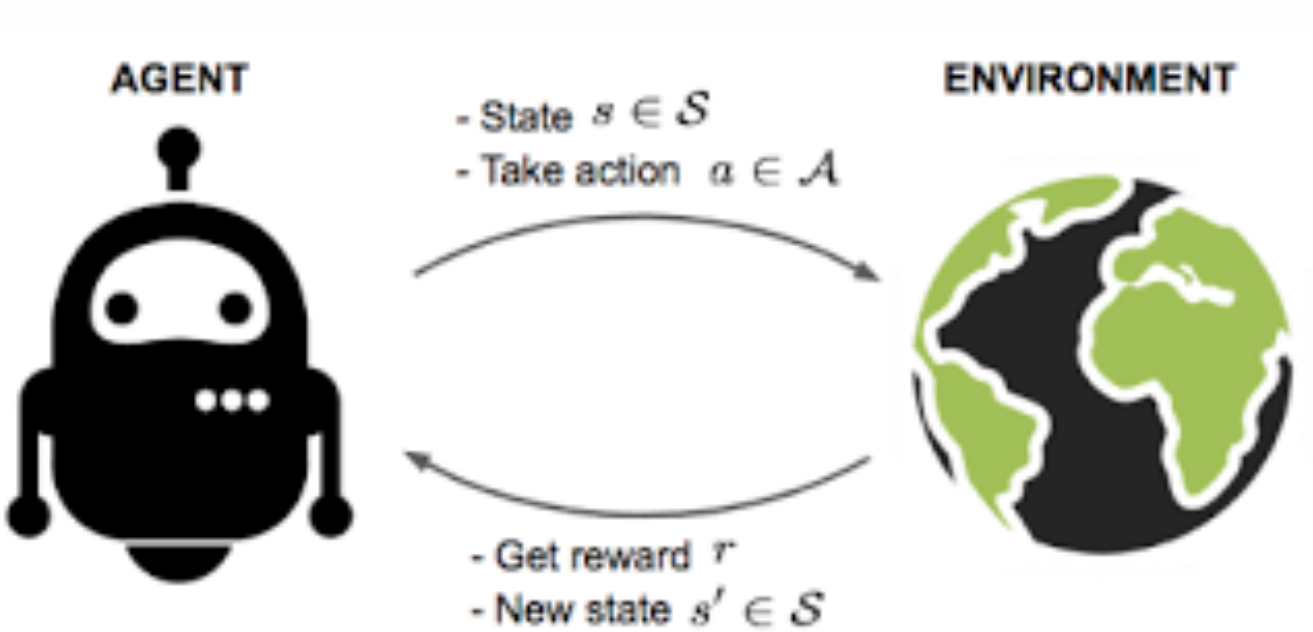
Reinforcement Learning: more abstract view

- **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: $s \xrightarrow{a} s' \quad (p(s' \mid s, a))$
- (immediate) reward $r(s, a, s')$

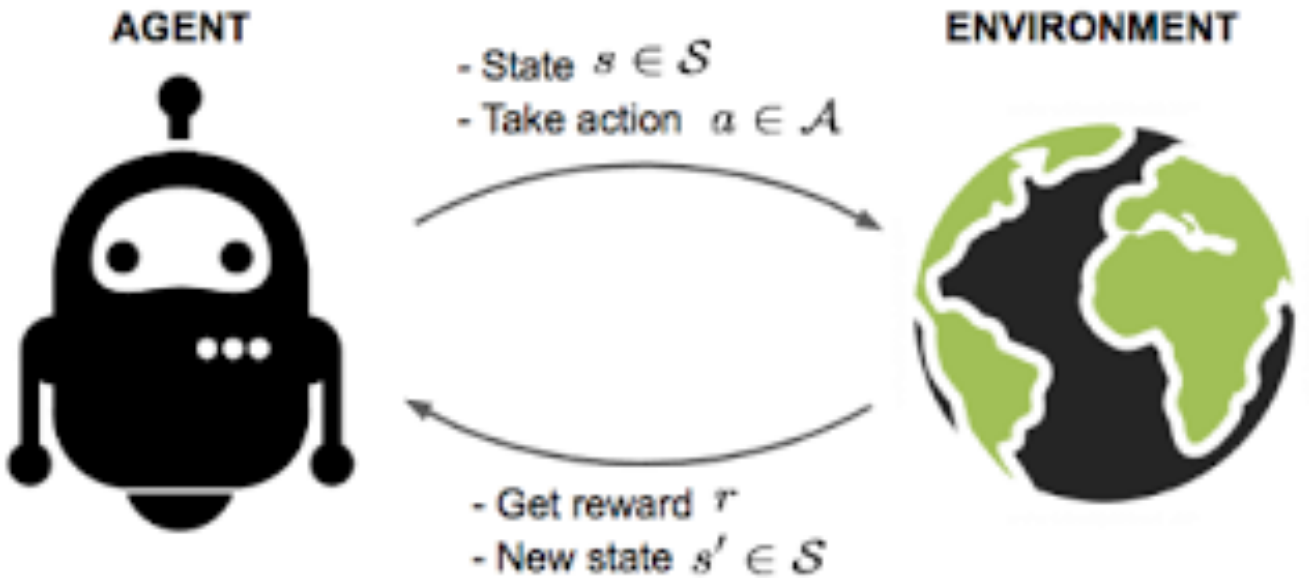


- **Example:** Taking a qualifying test (a), to pass from unqualified (s) to qualified state (s'), or not!

RL GOAL: Find optimal policy!

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

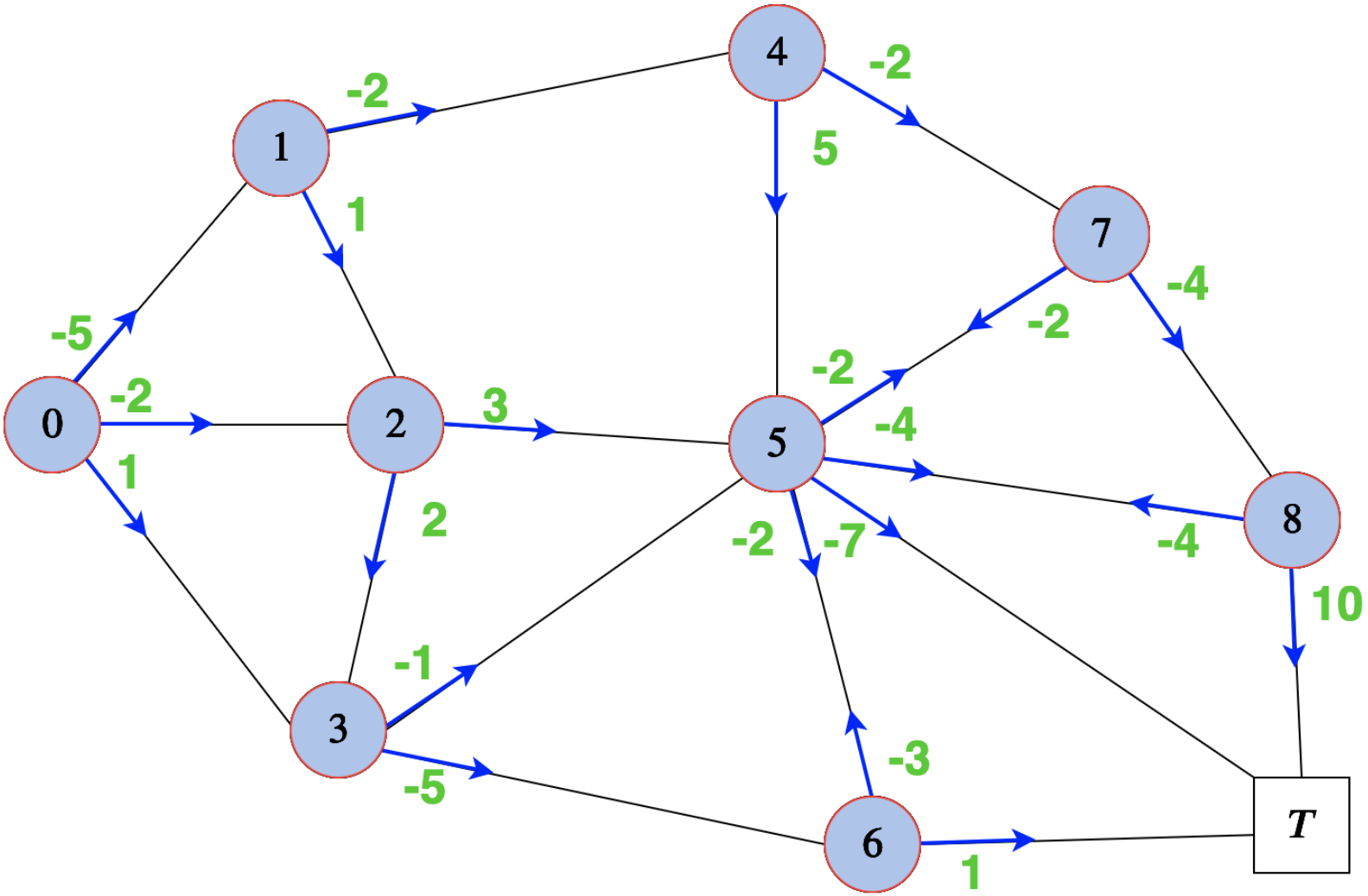
$$s \xrightarrow{\pi} a$$



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

Markov Decision Process (MDP)

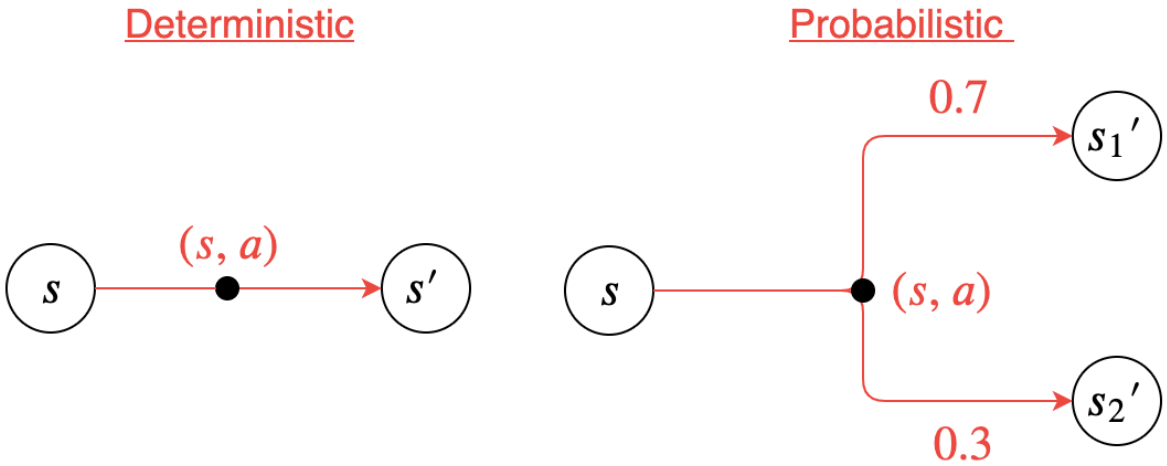
States + Actions + Transitions + Rewards



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** $r(s, a, s') = E[R \mid s, a, s']$: expected reward when taking action a at state s and transitioning to s'
    - Reward might depend on new state s' (e.g. $s' = \textit{pass}$ or $s' = \textit{fail}$)
  - A planning horizon H or **discount factor** γ;
    - How important are future rewards?
    - shortsighted $0 \leftarrow \gamma \rightarrow 1$ farsighted

Detailed Definition and Notation (1)



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

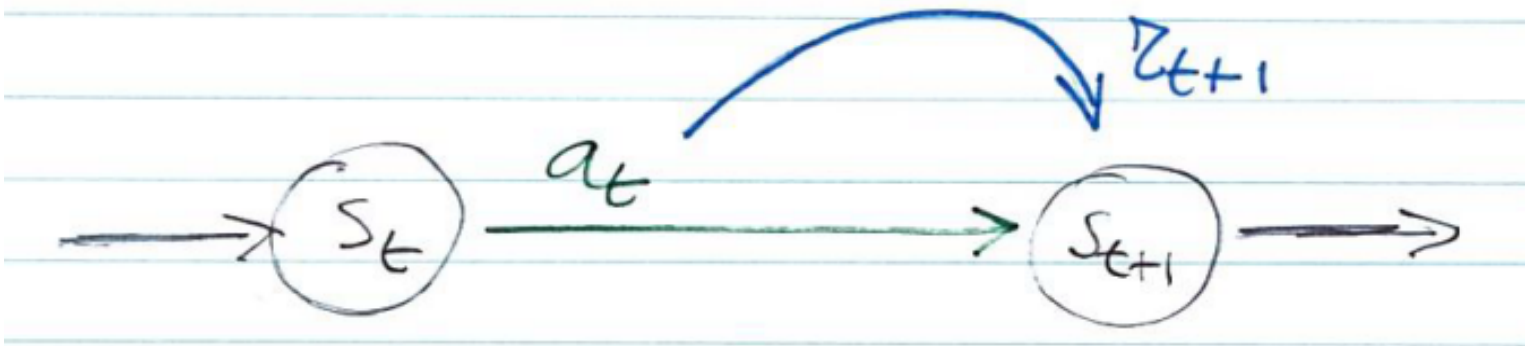
$p(s_1' \mid s, a) = 0.7$

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

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"*

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

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

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

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

Joint versus marginal distribution

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

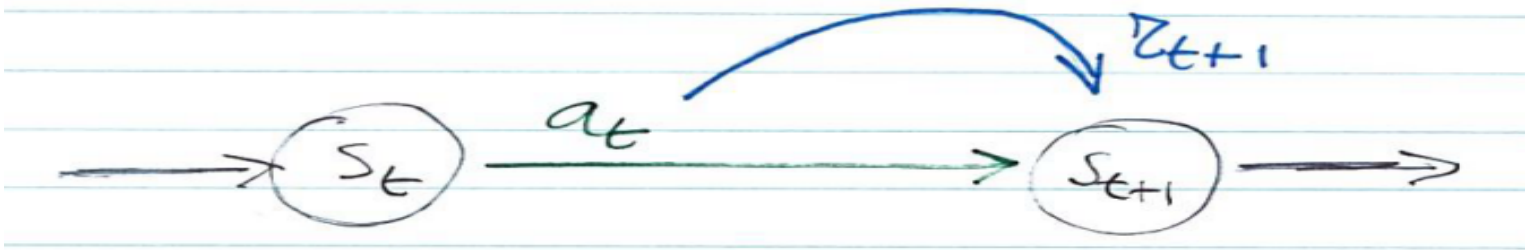
Dependence versus independence

- 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!

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:**

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

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

Return and Rewards

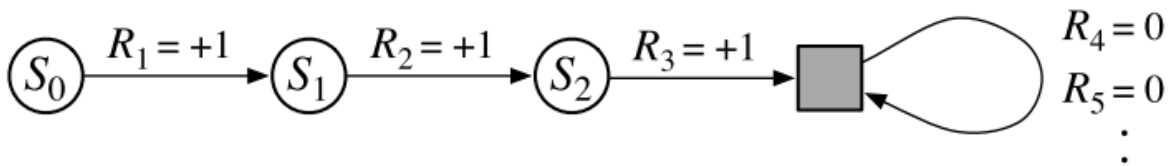
- The goal of the agent is to maximize the expected **(long-term) return**, a (discounted) sum over the immediate rewards received:

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

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:

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

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;

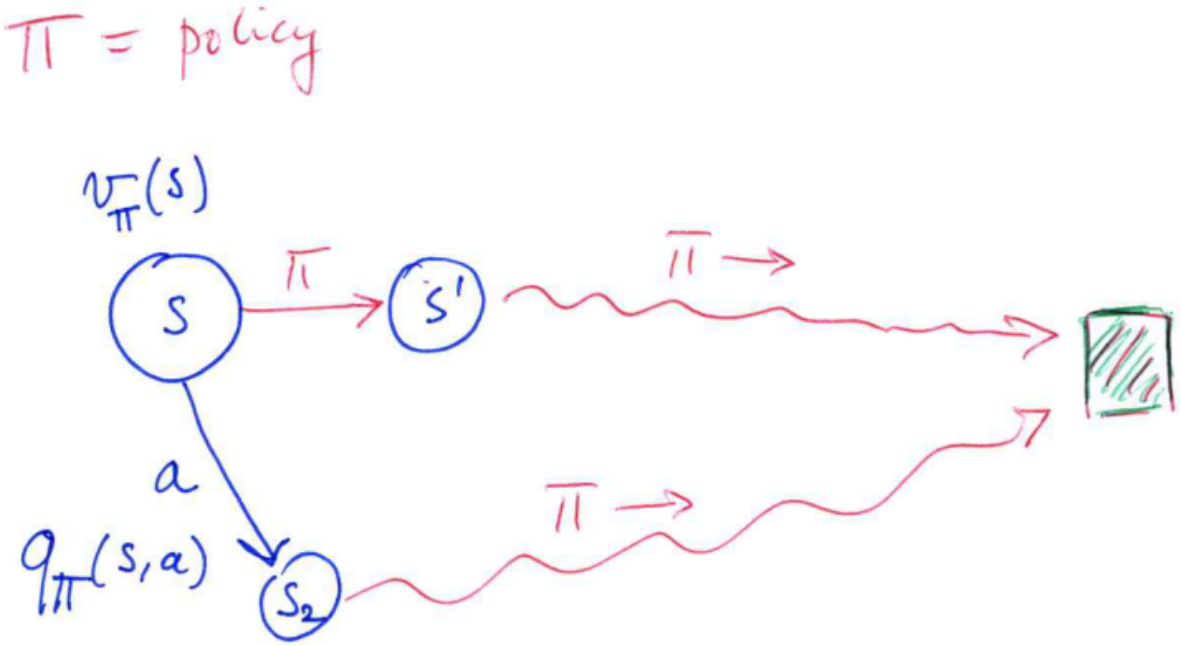
$$v_\pi(s) := E_\pi\Big[G_0 \,|\, S_0 = s\Big] = E_\pi\Big[\sum_{k=0}^{\infty} \gamma^k R_{k+1} | S_0 = s\Big]$$

- 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.

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:

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

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 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:

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

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:

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

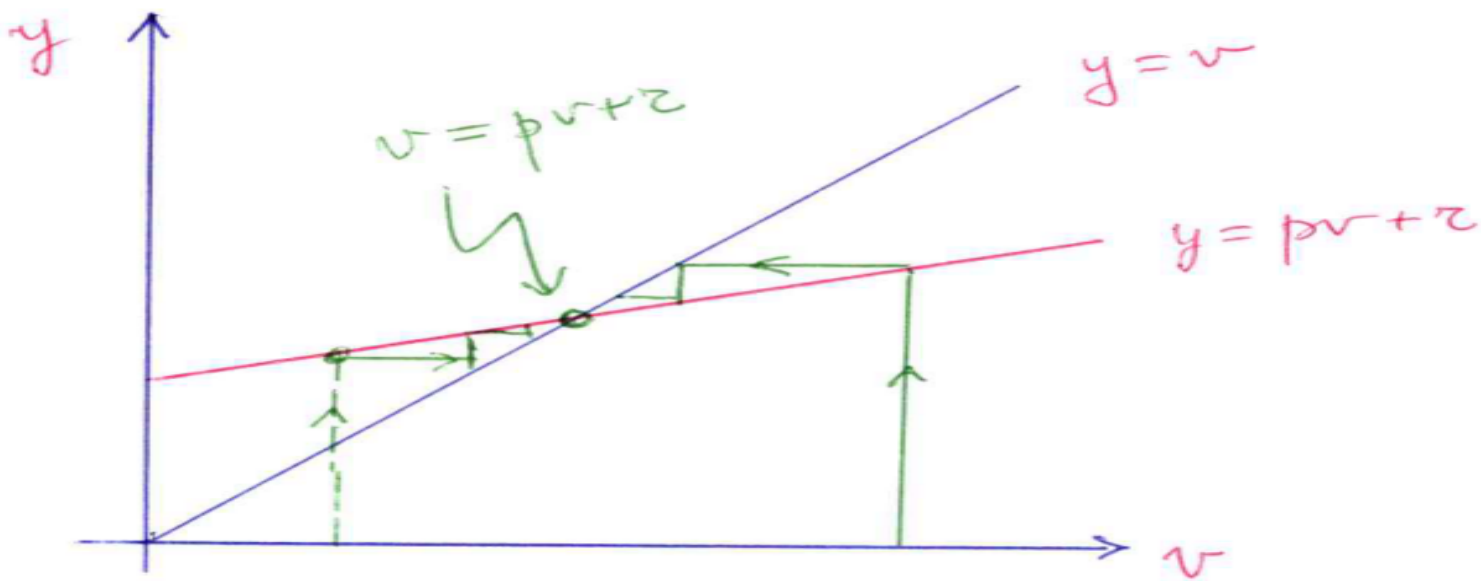
Solving Bellman eqs: Iteration to Fix-Point

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:

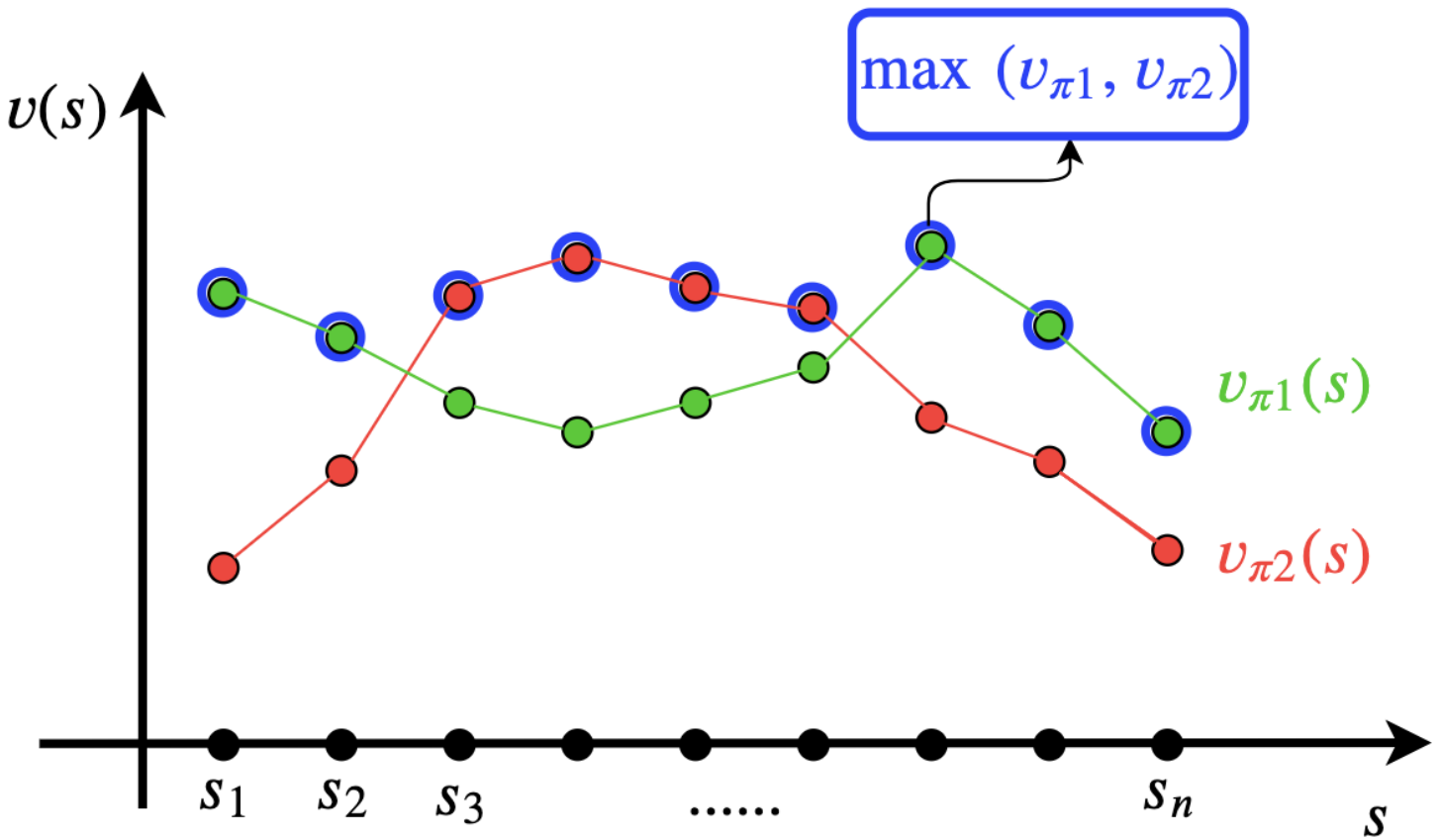
$$\mathbf{v}^{k+1} = \gamma P \mathbf{v}^k + \mathbf{r}$$



Optimal value functions

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

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

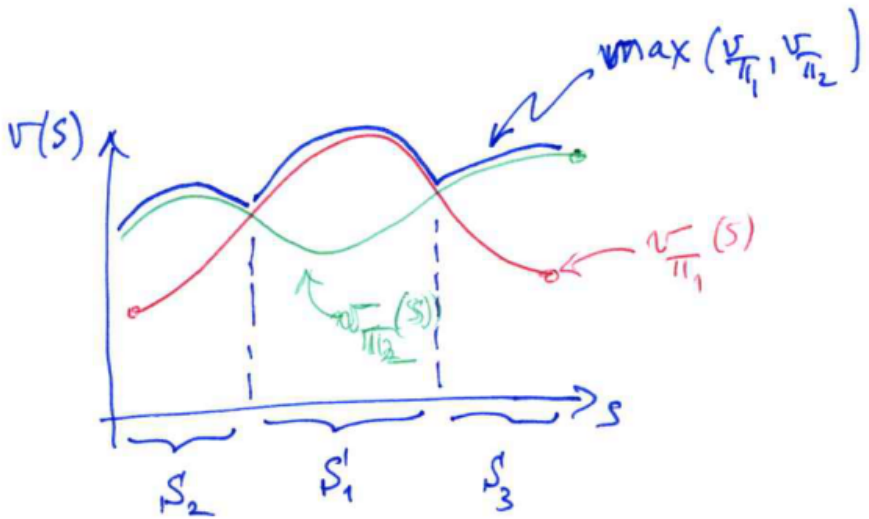


Optimal policy π^*

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

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

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

Optimal value functions: from weighted mean to max

- State-action value function
  - General:

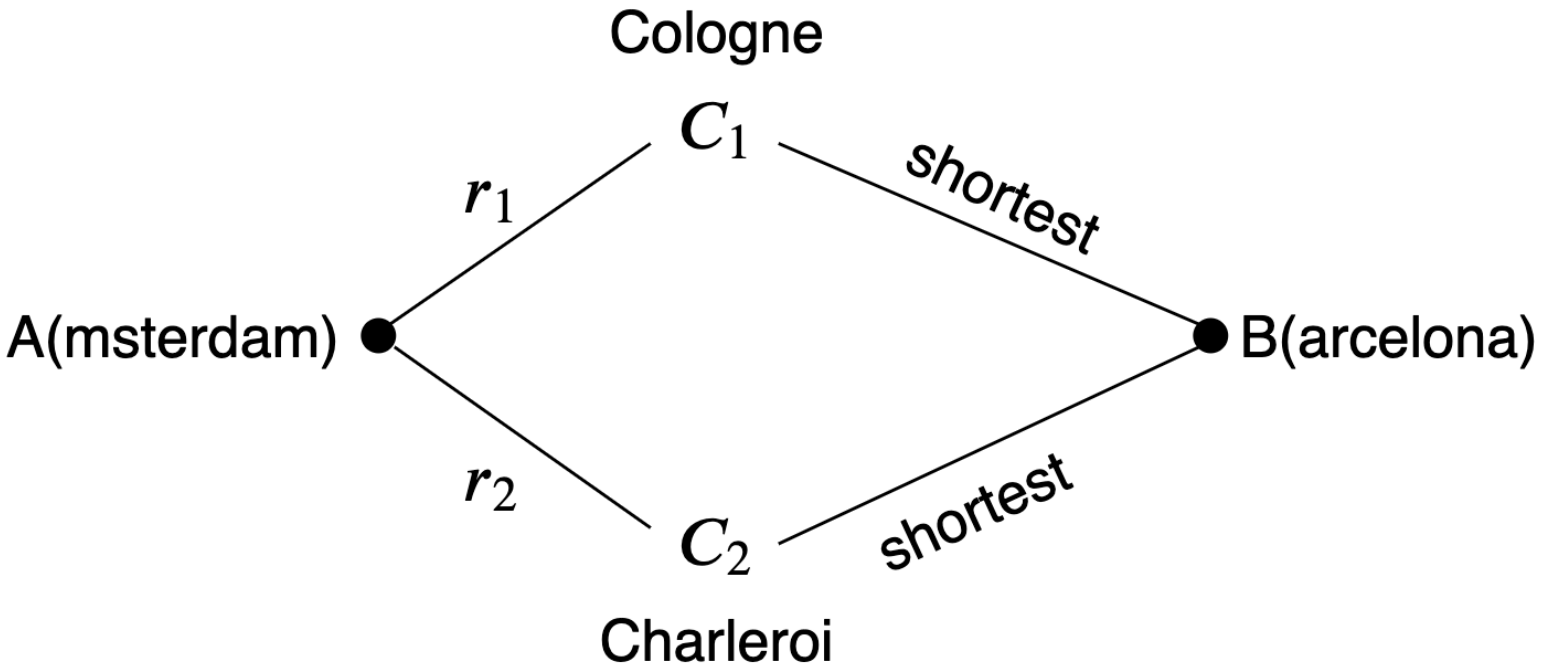
$$q_\pi(s,a) = \sum_{s'} p(s' \mid s,a)(r(s,a,s') + \gamma v_\pi(s))$$

  - 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!

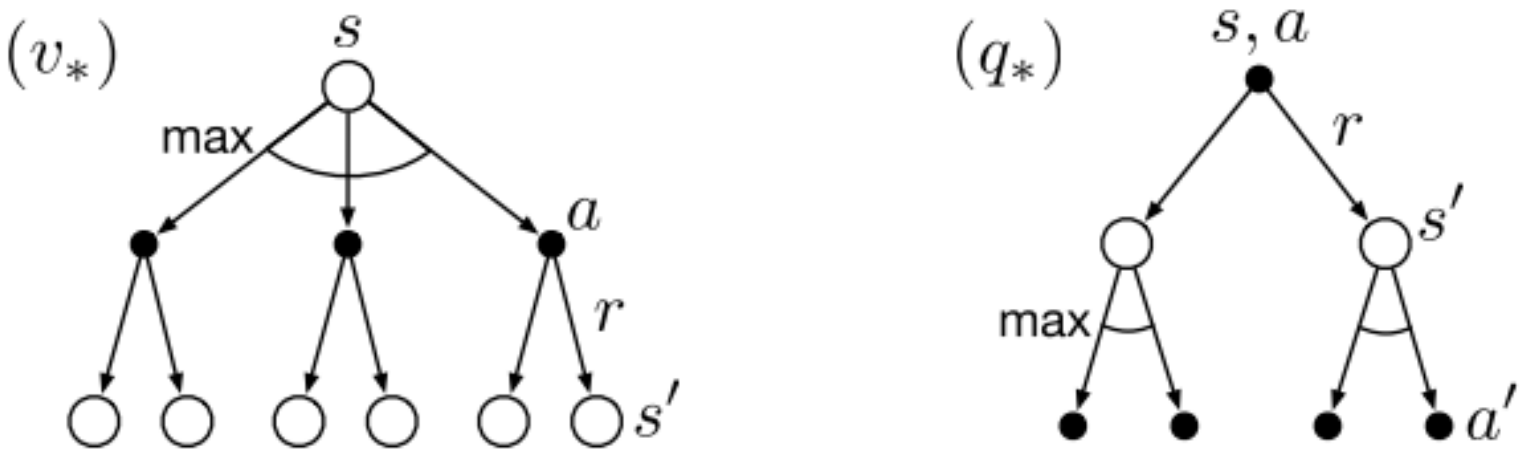


Figure 3.5: Backup diagrams for v_* and q_*

Bellman optimality equation in matrix form

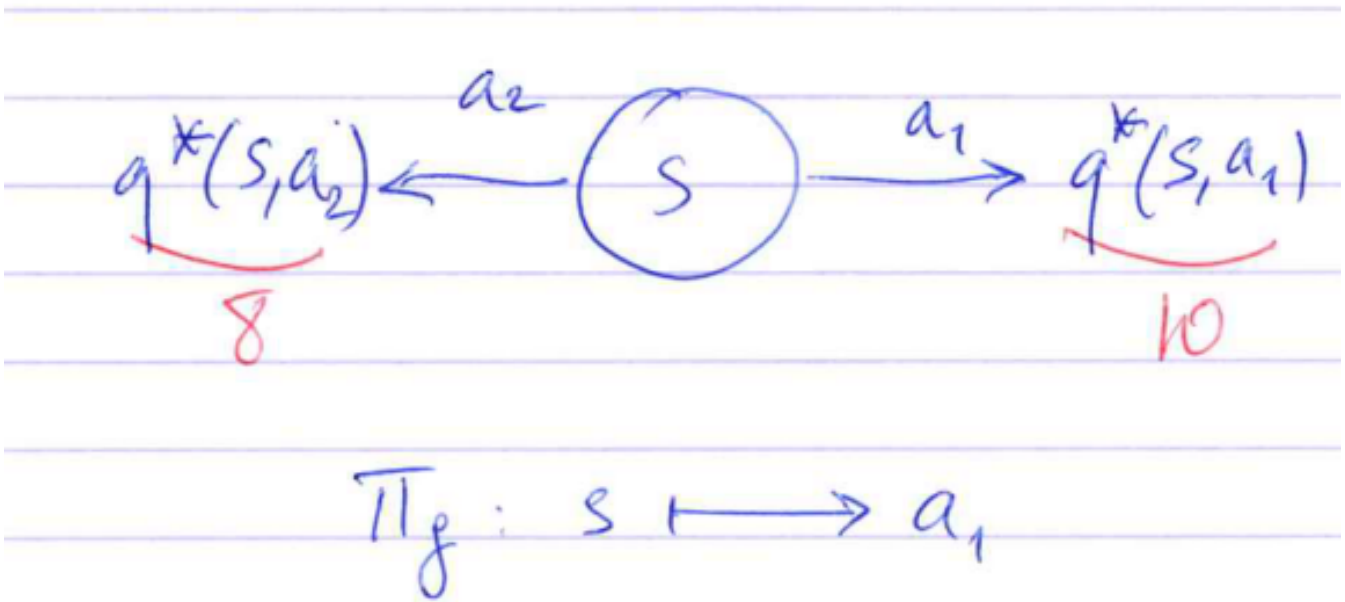
$$
\begin{aligned}
v^*(s) &= \max_{a \in A} \sum_{s'} p(s' \mid s, a) \Big[r(s, a, s') + \gamma v^*(s') \Big] \\
&= \max_a \left(R(s, a) + \gamma \sum_{s'} \underbrace{p(s' \mid s, a)}_{T_a(s,s')} v^*(s') \right) \\
&= \max_a \left(R(s, a) + \gamma \sum_{s'} T_a(s, s') v^*(s') \right)
\end{aligned}
$$

or in matrix notation:

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

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^*:

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

Bellman optimality equations

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

- **Optimal state-action value function**

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

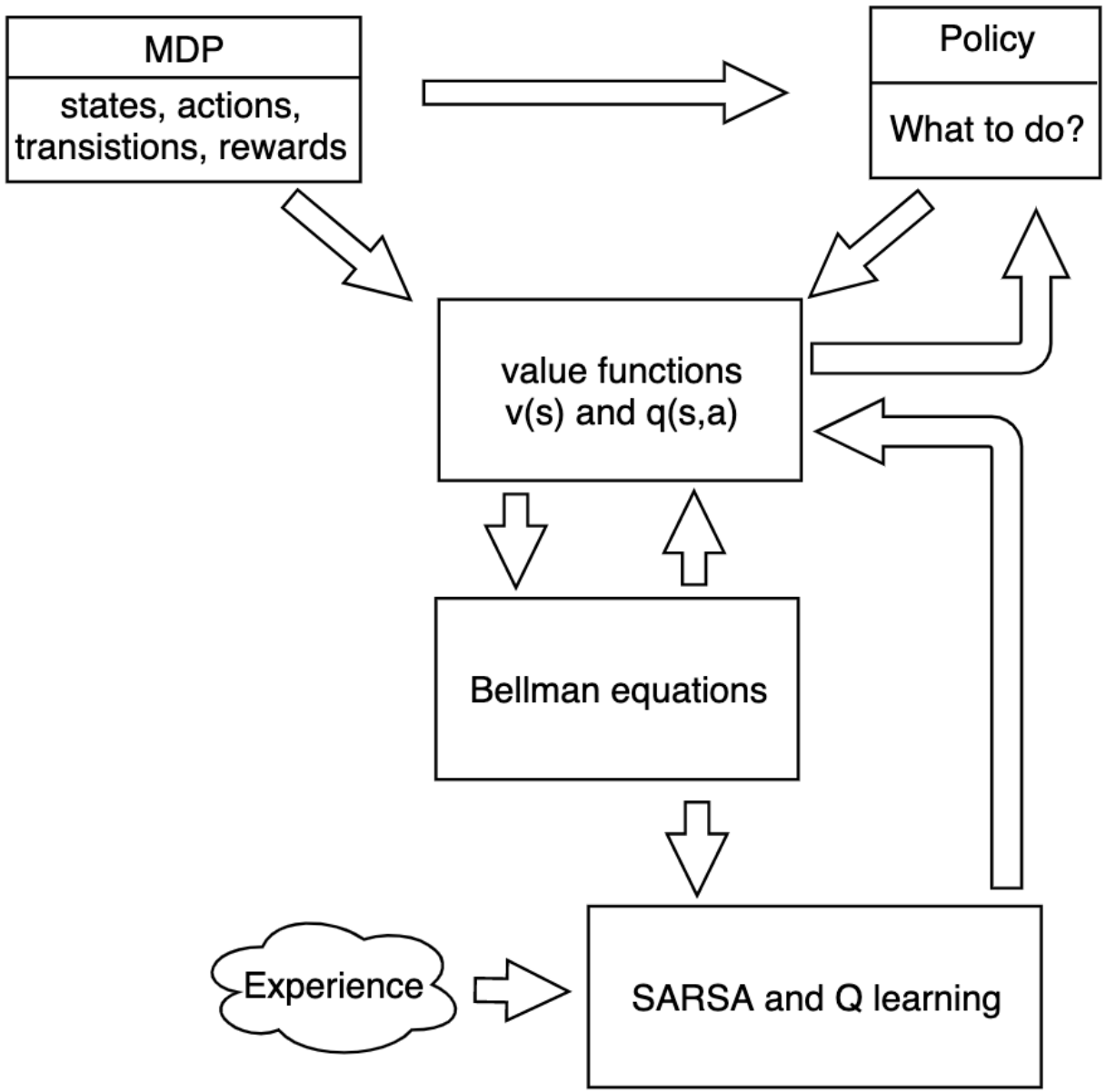
- **Recall: Optimal state value function**

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

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!

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"*