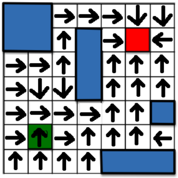
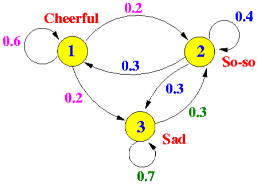
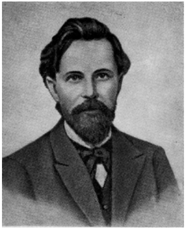
Machine Learning

Reinforcement Learning – Markov Decision Processes



Marcello Restelli



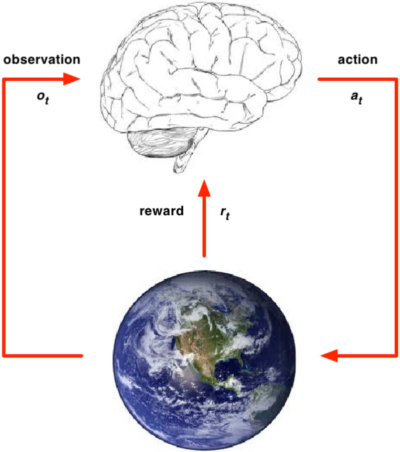


Outline

Sequential Decision Making

- **Goal**: select actions to maximize cumulative rewards
- Actions may have **long–term** consequences
- Reward may be **delayed**
- It may be better to **sacrifice** immediate reward to gain more long–term reward
- Examples:
  - A financial investment (may take months to mature)
  - Refueling a helicopter (might prevent a crash in several hours)
  - Blocking opponent moves (might help winning chances many moves from now)

Agent–Environment Interface



- At each step t the agent:
  - Executes action a_t
  - Receives observation o_t
  - Receives scalar reward r_t
- The environment:
  - Receives action a_t
  - Emits observation o_t
  - Emits scalar reward r_t

History and State

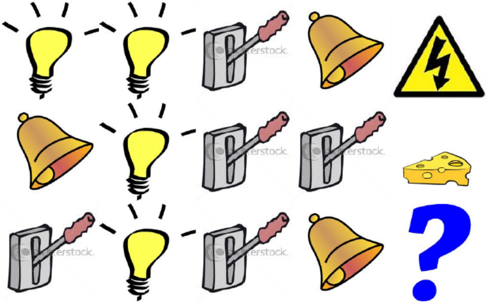
- The **history** is the sequence of observations, actions, rewards

$$h_t = a_1, o_1, r_1, \ldots, a_t, o_t, r_t$$

  - all observable variables up to time t
  - the **sensorimotor** stream of a robot or embodied agent
- What happens next depends on the history
  - agent selects actions
  - environment selects observations and rewards
- **State** is the information used to determine what happens next
- Formally, state is a function of the history:

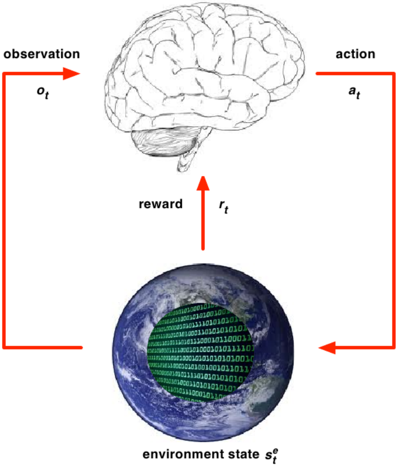
$$s_t = f(a_1, o_1, r_1, \ldots, a_t, o_t, r_t)$$

Rat Example



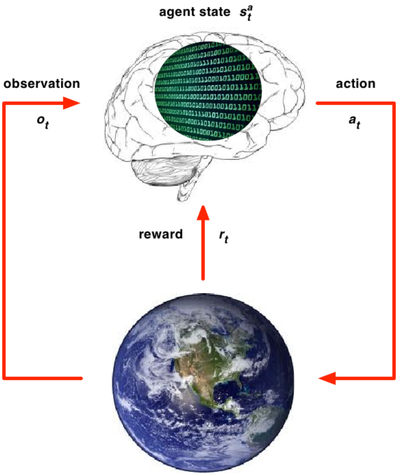
- What if agent state = last 3 observations ?
- What if agent state = counts of different observations?
- What if agent state = complete sequence?

Environment State



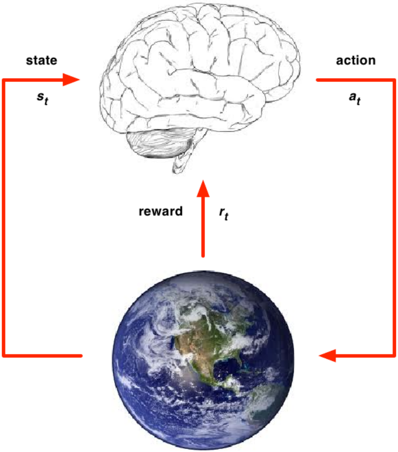
- The environment state s_t^e is the environment's private representation
  - whatever representation the environment uses to produce the next observation/reward
- The environment state is **not usually visible** to the agent
- Even if s_t^e is visible, it may contain **irrelevant** information

Agent State



- The agent state is the agent's internal representation
  - whatever information the agent uses to **select** the next action
  - is the information used by RL agents
- It can be any function history: $s_t^a = f(h_t)$

Fully Observable Environments



- **Full observability**: agent **directly** observes environment state

$$o_t = s_t^a = s_t^e$$

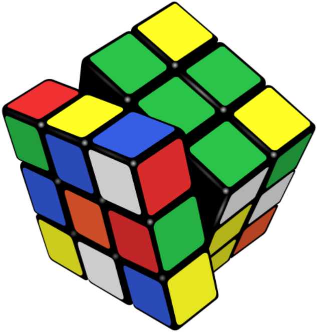
- Formally, this is a **Markov Decision Process** (MDP)
- The majority of this course will consider the MDP case

When is RL useful?

- When the dynamics of the environment are **unknown** or difficult to be modeled
  - e.g., trading, betting
- When the model of the environment is too **complex** to be solved exactly, so that **approximate** solutions are searched for
  - e.g., humanoid robot control, group elevator dispatching

Example 1: Rubik's Cube

- Invented in 1974 by Ernõ Rubik
- Formalization
  - State space: $\sim 4.33 \times 10^{19}$
  - Actions: 12 for each state
  - Deterministic state transitions
  - Rewards: -1 for each step
  - Undiscounted
- The cube can be solved in 20 moves or fewer

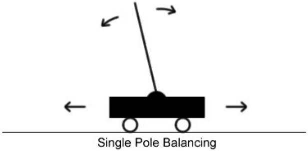


Example 2: Blackjack

- The most played casino game
- Formalization
  - State space: totals ~ 800, composition $\sim 104,000$
  - Actions: from 2 to 4 according to the state
  - Stochastic state transitions
  - Rewards: 0 for each step, $\{-2, -1, 0, 1, 1.5, 2\}$ at the end
  - Undiscounted
- Using the optimal policy, the house edge is very low ($\sim 0,4\% - 0,7\%$)

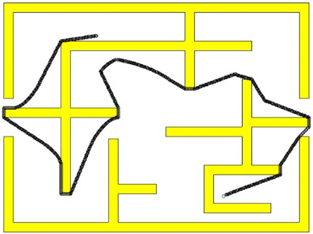
Example 3: Pole balancing

- A classical RL benchmark
- Formalization
  - State space: four continuous state variables $x, \dot{x}, \theta, \dot{\theta}$
  - Actions: two actions $\{-N, N\}$
  - Deterministic state transitions
  - Rewards:
    - 0 when in the goal region
    - -1 when outside goal region
    - -100 when outside feasible region



Example 4: Robot Navigation

- The most important task in mobile robotics
- Formalization
  - State space: robot coordinates
  - Actions: moving actions
  - Stochastic state transitions
  - Rewards: -1 until goal is reached
  - Undiscounted/discounted
- Often the state cannot be observed
- Shift to POMDP framework



Example 5: Web Banner Advertising

- We have to choose which banner ad showing in a certain slot of our web page
- Formalization
  - State space: single state or multiple states (contexts)
  - Actions: one for each banner
  - No dynamics
  - Rewards: probability of click times the cost per click
- Multi-armed bandit: exploration vs exploitation

Example 6: Chess

- Very popular board game
- Formalization
  - State space: $\sim 10^{47}$
  - Actions: from 0 to 218
  - Deterministic opponent-dependent state transitions
  - Rewards: 0 each step, $\{-1, 0, 1\}$ at the end
  - Undiscounted
- The size of the game tree is 10^{123}

Example 7: Texas Hold'em

- Recently, the most played poker version
- Formalization
  - State space: huge, and not observable
  - Actions: fold, call, and raise
  - Stochastic opponent-dependent state transitions
  - Rewards: 0 each step, $\{-1, 0, 1\}$ at the end
  - Undiscounted
- The size of the limit game with 2 players is 10^{18}

Markov Assumption

"The future is independent of the past given the present"

Definition

A stochastic process X_t is said to be **Markovian** if and only if

$$\mathbb{P}(X_{t+1} = j | X_t = i, X_{t-1} = k_{t-1}, \dots, X_1 = k_1, X_0 = k_0) = \mathbb{P}(X_{t+1} = j | X_t = i)$$

- The state **captures all the information** from history
- Once the state is known, the history may be **thrown away**
- The state is a **sufficient statistic** for the future
- The conditional probabilities are **transition probabilities**
- If the probabilities are **stationary** (time invariant), we can write:

$$p_{ij} = \mathbb{P}(X_{t+1} = j | X_t = i) = \mathbb{P}(X_1 = j | X_0 = i)$$

Discrete–time Finite Markov Decision Processes

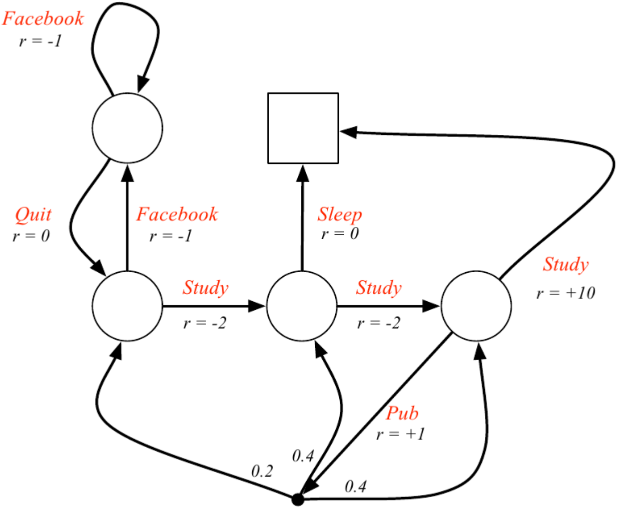
A Markov decision process (MDP) is Markov reward process with **decisions**. It models an environment in which all states are Markov and time is divided into **stages**.

Definition

A **Markov Process** is a tuple $\langle \mathcal{S}, \mathcal{A}, P, R, \gamma, \mu \rangle$

- $\mathcal{S}$ is a (finite) set of states
- $\mathcal{A}$ is a (finite) set of actions
- P is a state transition probability matrix, $P(s'|s, a)$
- R is a reward function, $R(s, a) = \mathbb{E}[r|s, a]$
- γ is a discount factor, $\gamma \in [0, 1]$
- a set of initial probabilities $\mu_i^0 = P(X_0 = i)$ for all i

Example: Student MDP



Goals and Rewards

- Is a scalar **reward** an adequate notion of a **goal**?
    - **Sutton hypothesis**: That all of what we mean by goals and purposes can be well thought of as the maximization of the cumulative sum of a received scalar signal (reward)
    - Probably ultimately wrong, but so **simple** and **flexible** we have to disprove it before considering anything more complicated
- A goal should specify **what** we want to achieve, not **how** we want to achieve it
- The same goal can be specified by (infinite) **different reward functions**
- A goal must be outside the agent's direct control – thus outside the agent
- The agent must be able to measure success:
    - **explicitly**
    - **frequently** during her lifespan

Return

- Time horizon
  - **finite**: finite and fixed number of steps
  - **indefinite**: until some stopping criteria is met (**absorbing** states)
  - **infinite**: forever
- Cumulative reward
  - total reward:
$$V = \sum_{i=1}^{\infty} r_i$$
  - average reward:
$$V = \lim_{n \to \infty} \frac{r_1 + \cdots + r_n}{n}$$
  - discounted reward:
$$V = \sum_{i=1}^{\infty} \gamma^{i-1} r_i$$
  - mean–variance reward

Infinite–horizon Discounted Return

Definition

The **return** v_t is the total discounted reward from time–step t.

$$v_t = r_{t+1} + \gamma r_{t+2} + \cdots = \sum_{k=0}^{\infty} \gamma^k r_{t+k+1}$$

- The discount $\gamma \in [0, 1)$ is the present value of future rewards
- The value of receiving reward r after $k + 1$ time–steps is $\gamma^k r$
- **Immediate** reward vs **delayed** reward
  - γ close to 0 leads to "**myopic**" evaluation
  - γ close to 1 leads to "**far–sighted**" evaluation
- γ can be also interpreted as the **probability** that the process will **go on**

Why discount?

Most Markov reward (and decision) processes are discounted, why?

- **Mathematically** convenient to discount rewards
- **Avoids infinite returns** in cyclic Markov processes
- **Uncertainty** about the future may not be fully represented
- If the reward is **financial**, immediate rewards may earn more interest than delayed rewards
- **Animal/human behavior** shows preference for immediate reward
- It is sometimes possible to use **undiscounted** Markov reward processes (i.e. $\gamma = 1$), e.g. if all sequences **terminate**

Policies

- A policy, at any given point in time, **decides** which action the agent selects
- A policy fully defines the **behavior** of an agent
- Policies can be:
  - Markovian $\subseteq$ History–dependent
  - Deterministic $\subseteq$ Stochastic
  - Stationary $\subseteq$ Non–stationary

Stationary Stochastic Markovian Policies

Definition

A policy π is a **distribution over actions** given the state:

$$\pi(a|s) = \mathbb{P}[a|s]$$

- MDP policies depend on the **current state** (not the history)
- i.e., Policies are **stationary** (time–independent)
- Given an MDP $\mathcal{M} = \langle \mathcal{S}, \mathcal{A}, P, R, \gamma, \mu \rangle$ and a policy π
  - The state sequence $s_1, s_2, \dots$ is a **Markov process** $\langle \mathcal{S}, P^\pi, \mu \rangle$
  - The state and reward sequence $s_1, r_2, s_2, \dots$ is a **Markov reward process** $\langle \mathcal{S}, P^\pi, R^\pi, \gamma, \mu \rangle$, where

$$P^\pi = \sum_{a \in \mathcal{A}} \pi(a|s) P(s'|s,a) \quad R^\pi = \sum_{a \in \mathcal{A}} \pi(a|s) R(s,a)$$

Value Functions

Given a policy π, it is possible to define the **utility** of each state: **Policy Evaluation**

Definition

The state–value function $V^{\pi}(s)$ of an MDP is the expected return starting from state s, and then following policy π

$$V^{\pi}(s) = \mathbb{E}_{\pi}[v_t | s_t = s]$$

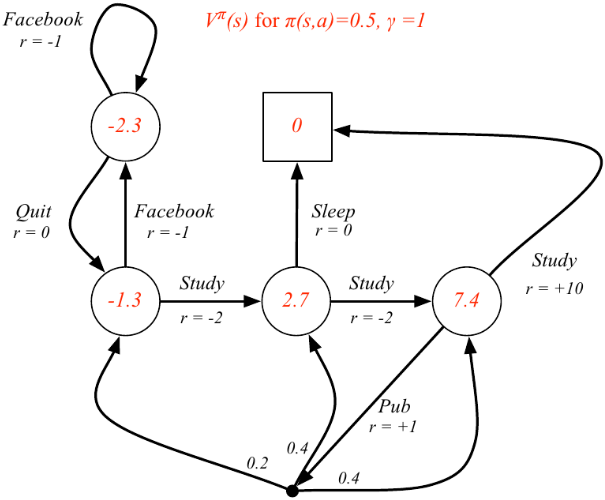
For **control purposes**, rather than the value of each state, it is easier to consider **the value of each action** in each state

Definition

The action–value function $Q^{\pi}(s, a)$ is the expected return starting from state s, taking action a, and then following policy π

$$Q^{\pi}(s, a) = \mathbb{E}_{\pi}[v_t | s_t = s, a_t = a]$$

Example: Value Function of Student MDP



Bellman Expectation Equation

The state–value function can again be **decomposed** into immediate reward plus discounted value of successor state,

$$\begin{aligned} V^\pi(s) &= \mathbb{E}_\pi[r_{t+1} + \gamma V^\pi(s_{t+1}) | s_t = s] \\ &= \sum_{a \in A} \pi(a|s) \left(R(s,a) + \gamma \sum_{s' \in S} P(s'|s,a) V^\pi(s') \right) \end{aligned}$$

The action-value function can similarly be decomposed

$$\begin{aligned} Q^\pi(s,a) &= \mathbb{E}_\pi[r_{t+1} + \gamma Q^\pi(s_{t+1}, a_{t+1}) | s_t = s, a_t = a] \\ &= R(s,a) + \gamma \sum_{s' \in S} P(s'|s,a) V^\pi(s') \\ &= R(s,a) + \gamma \sum_{s' \in S} P(s'|s,a) \sum_{a' \in A} \pi(a'|s') Q^\pi(s',a') \end{aligned}$$

Bellman Expectation Equation (Matrix Form)

The Bellman expectation equation can be expressed **concisely** using the induced MRP

$$V^\pi = R^\pi + \gamma P^\pi V^\pi$$

with **direct solution**

$$V^\pi = (I - \gamma P^\pi)^{-1} R^\pi$$

Bellman operators for V^π

Definition

The Bellman operator for V^π is defined as $T^\pi : \mathbb{R}^{|\mathcal{S}|} \to \mathbb{R}^{|\mathcal{S}|}$ (maps value functions to value functions):

$$(T^\pi V^\pi)(s) = \sum_{a \in A} \pi(a|s) \left(R(s,a) + \gamma \sum_{s' \in S} P(s'|s,a) V^\pi(s') \right)$$

- Using Bellman operator, Bellman expectation equation can be **compactly** written as:
$$T^\pi V^\pi = V^\pi$$
- V^π is a **fixed point** of the Bellman operator T^π
- This is a **linear equation** in V^π and T^π
- If $0 < \gamma < 1$ then T^π is a **contraction** w.r.t. the maximum norm

Bellman operators for Q^π

Definition

The Bellman operator for Q^π is defined as $T^\pi : \mathbb{R}^{|\mathcal{S}|\times|\mathcal{A}|} \to \mathbb{R}^{|\mathcal{S}|\times|\mathcal{A}|}$ (maps action–value functions to action–value functions):

$$(T^\pi Q^\pi)(s,a) = R(s,a) + \gamma \sum_{s'\in S} P(s'|s,a) \sum_{a'\in A} \pi(a'|s') \left(Q^\pi(s',a')\right)$$

- Using Bellman operator, Bellman expectation equation can be compactly written as:

$$T^\pi Q^\pi = Q^\pi$$

- Q^π is a fixed point of the Bellman operator T^π
- This is a linear equation in Q^π and T^π
- If $0 < \gamma < 1$ then T^π is a contraction w.r.t. the maximum norm

Optimal Value Function

Definition

The **optimal state–value function** $V^*(s)$ is the maximum value function over all policies

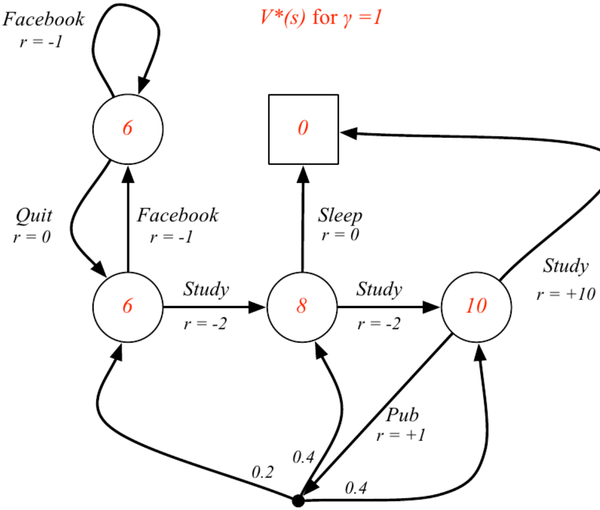
$$V^*(s) = \max_{\pi} V^{\pi}(s)$$

The **optimal action–value function** $Q^*(s, a)$ is the maximum action–value function over all policies

$$Q^*(s, a) = \max_{\pi} Q^{\pi}(s, a)$$

- The optimal value function specifies the **best** possible performance in the MDP
- An MDP is "**solved**" when we know the optimal value function

Example: Optimal Value Function of Student MDP



Optimal Policy

Value functions define a partial ordering over policies

$$\pi \geq \pi' \text{ if } V^{\pi}(s) \geq V^{\pi'}(s), \quad \forall s \in \mathcal{S}$$

Theorem

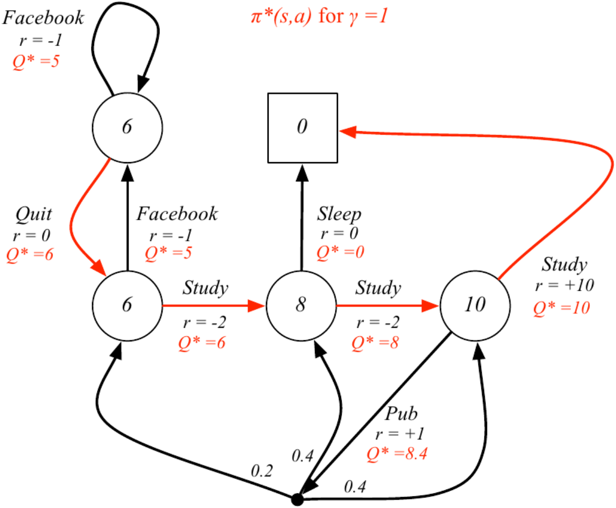
For any Markov Decision Process

- *There exists* ***an optimal policy*** π^* *that is better than or equal to all other policies* $\pi^* \geq \pi, \quad \forall \pi$
- ***All*** *optimal policies achieve the* ***optimal value function***, $V^{\pi^*}(s) = V^*(s)$
- ***All*** *optimal policies achieve the* ***optimal action–value function***, $Q^{\pi^*}(s, a) = Q^*(s, a)$
- *There is always a* ***deterministic optimal policy*** *for any MDP*

A deterministic optimal policy can be found by maximizing over $Q^*(s, a)$

$$\pi^*(a|s) = \begin{cases} 1 & \text{if } a = arg \max_{a \in \mathcal{A}} Q^*(s, a) \\ 0 & \text{otherwise} \end{cases}$$

Example: Optimal Policy for Student MDP



Bellman Optimality Equation

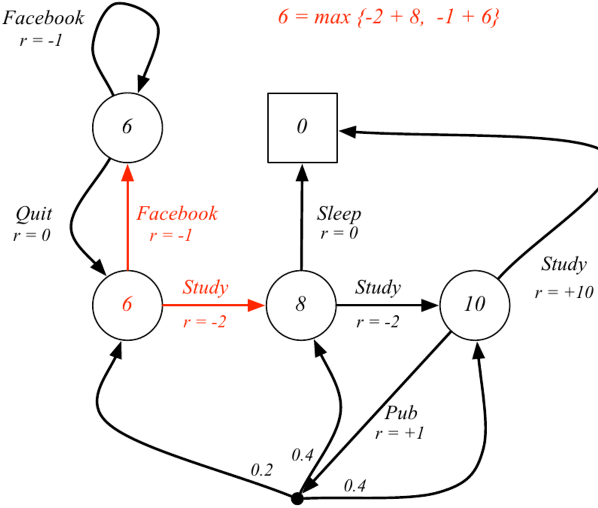
Bellman Optimality Equation for $V*$

$$\begin{aligned} V^*(s) &= \max_a Q^*(s,a) \\ &= \max_a \left\{ R(s,a) + \gamma \sum_{s' \in \mathcal{S}} P(s'|s,a) V^*(s') \right\} \end{aligned}$$

Bellman Optimality Equation for $Q*$

$$\begin{aligned} Q^*(s,a) &= R(s,a) + \gamma \sum_{s' \in \mathcal{S}} P(s'|s,a) V^*(s') \\ &= R(s,a) + \gamma \sum_{s' \in \mathcal{S}} P(s'|s,a) \max_{a'} Q^*(s',a') \end{aligned}$$

Example: Bellman Optimality Equation in Student MDP



Bellman Optimality Operator

Definition

The Bellman optimality operator for V^* is defined as $T^* : \mathbb{R}^{|\mathcal{S}|} \to \mathbb{R}^{|\mathcal{S}|}$ (maps value functions to value functions):

$$(T^* V^*)(s) = \max_{a \in A} \left(R(s,a) + \gamma \sum_{s' \in S} P(s'|s,a) V^*(s') \right)$$

Definition

The Bellman optimality operator for Q^* is defined as $T^* : \mathbb{R}^{|\mathcal{S}| \times |\mathcal{A}|} \to \mathbb{R}^{|\mathcal{S}| \times |\mathcal{A}|}$ (maps action–value functions to action–value functions):

$$(T^* Q^*)(s,a) = R(s,a) + \gamma \sum_{s' \in \mathcal{S}} P(s'|s,a) \max_{a'} Q^*(s',a')$$

Properties of Bellman Operators

- **Monotonicity**: if $f_1 \leq f2$ component–wise

$$T^\pi f_1 \leq T^\pi f_2 \quad , \quad T^* f_1 \leq T^* f_2$$

- **Max–Norm Contraction**: for two vectors f_1 anf f_2

$$\|T^\pi f_1 - T^\pi f_2\|_\infty \leq \gamma \|f_1 - f_2\|_\infty$$

$$\|T^* f_1 - T^* f_2\|_\infty \leq \gamma \|f_1 - f_2\|_\infty$$

- V^π is the **unique fixed point** of T^π
- V^* is the **unique fixed point** of T^*
- For any vector $f \in \mathbb{R}^{|\mathcal{S}|}$ and any policy π, we have

$$\lim_{k \to \infty} (T^\pi)^k f = V^\pi \quad , \quad \lim_{k \to \infty} (T^*)^k f = V^*$$

Solving the Bellman Optimality Equation

- Bellman optimality equation is **non–linear**
- **No closed form** solution for the general case
- Many **iterative** solution methods
  - Dynamic Programming
    - Value Iteration
    - Policy Iteration
  - Linear Programming
  - Reinforcement Learning
    - Q–learning
    - SARSA

Extensions to MDP

- Undiscounted, average reward MDPs
- Infinite and continuous MDPs
- Partially observable MDPs
- Semi-MDPs
- Non–stationary MDPs, Markov games