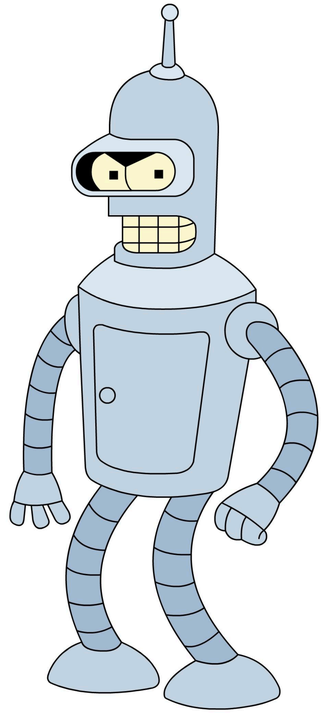
Reinforcement Learning

HSE, winter - spring 2025

Lecture 1: Introduction to RL



Sergei Laktionov
slaktionov@hse.ru
LinkedIn

Course Summary

- 10 weeks (lecture + practical seminar)
- 5 home assignments (2 simple + 3 hard) (0.65)
- 1 paper’s replication and presentation (0.25)
- 9 quizzes before each class (0.1)

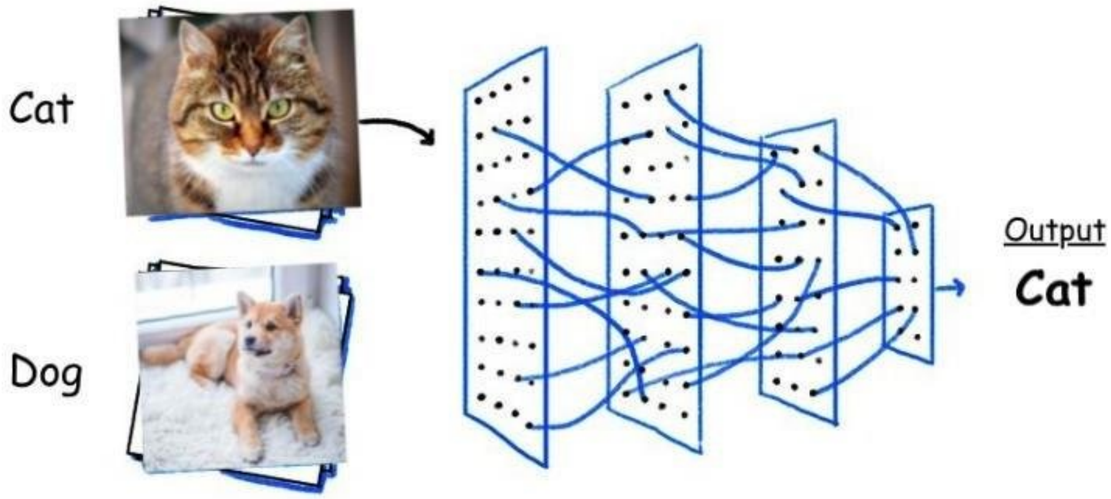
Introduction

Recap: Supervised Learning

Train sample: $(x_i, y_i)_{i=1}^n \sim i\,.\,i\,.\,d\,.$

Approximation: $y_i \approx \hat{y}_i = f(x_i)$, where
$f \in \mathcal{F}$, $\mathcal{F}$ is a family of algorithms

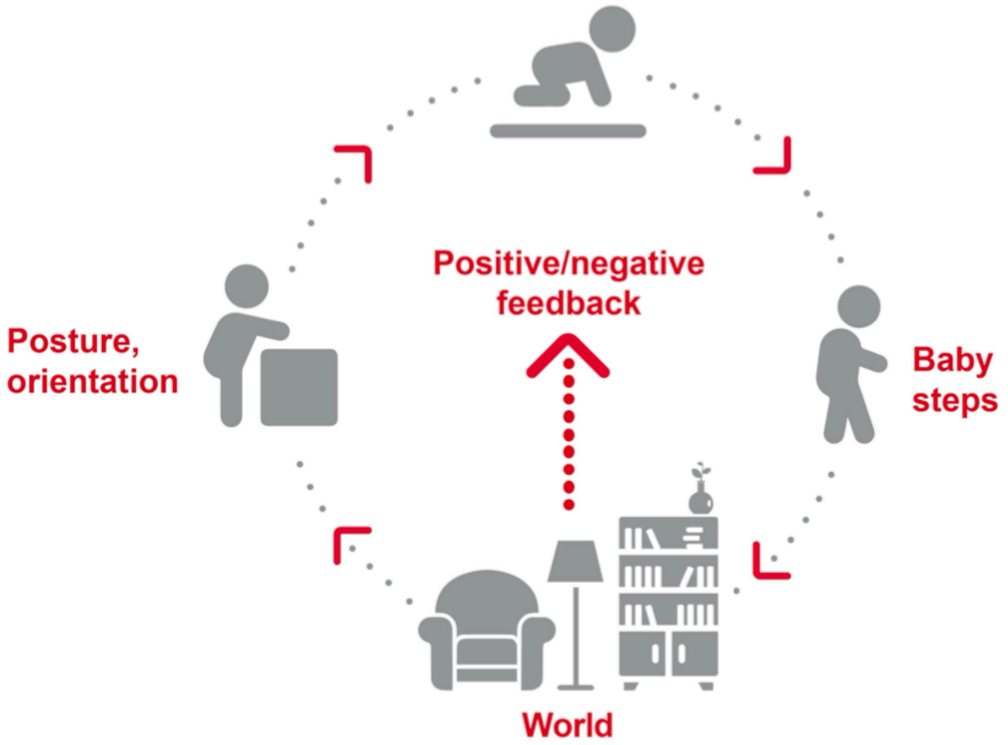
Loss minimisation: $\frac{1}{n}\sum_{i=1}^{n} L(f(x_i), y_i) \to \min_{f \in \mathcal{F}}$



Source

Sequential Decision Making

Baby learning



Source

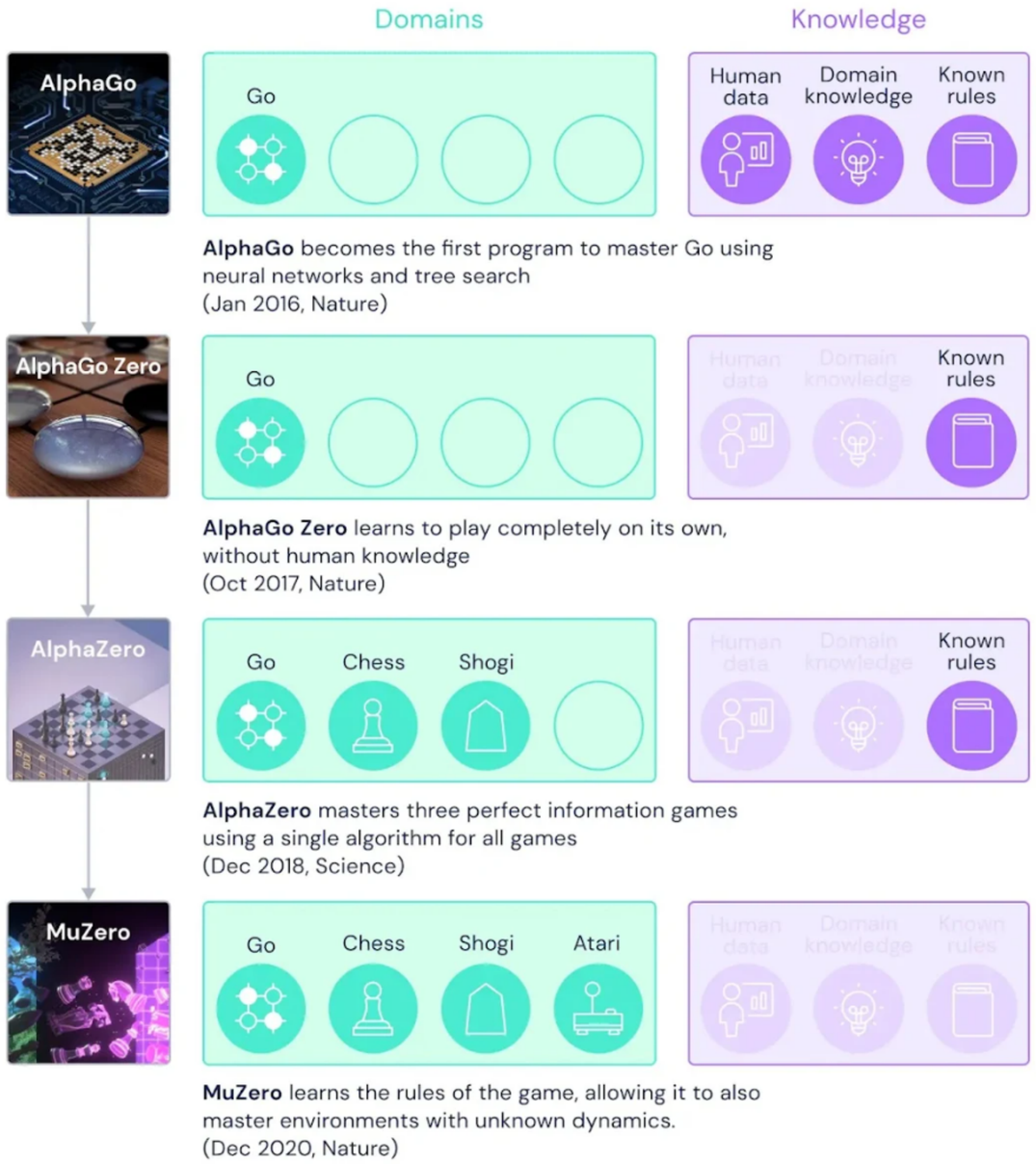
Self-driving car

Source

Characteristics of RL

- There is no supervisor, only a reward signal
- Feedback is delayed, not instantaneous
- Time really matters (sequential, non i.i.d. data)
- Agent’s actions affect the subsequent data it receives

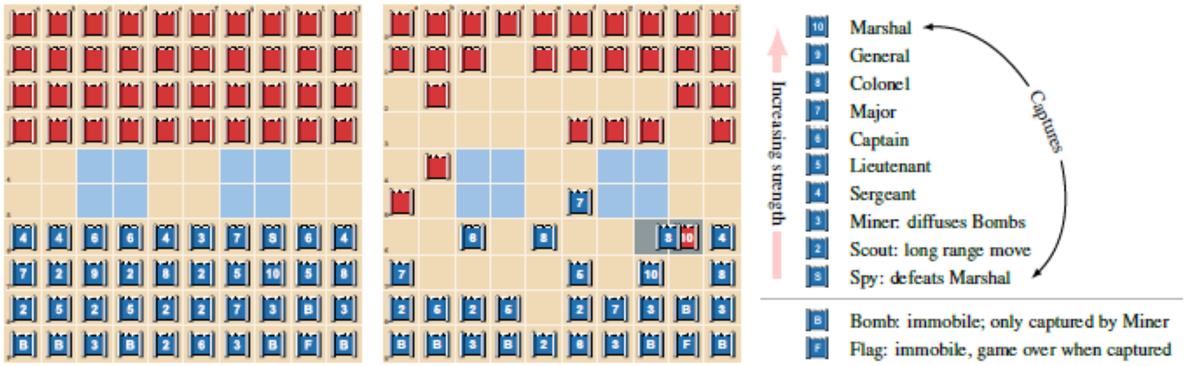
Examples: Board Games



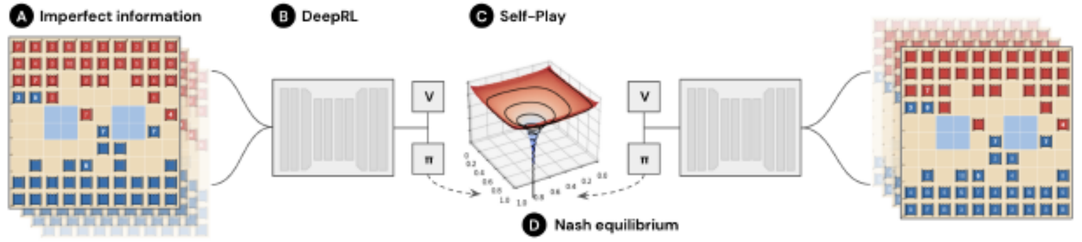
https://deepmind.google/discover/blog/muzero-mastering-go-chess-shogi-and-atari-without-rules/

Examples: Board Games with Imperfect Information

https://arxiv.org/abs/2206.15378



(a) Stratego is a two-player board game where each player aims to capture the opponent's flag. To do so, they each have 40 pieces of diverse strengths. The game starts with the deployment phase, where both players secretly position their pieces on the board. In a second game-play phase, the players take turns moving pieces. When two pieces are in the same location, they are revealed, and the weaker piece is removed, or both if they have the same strength. When the weakest movable piece, the Spy, attacks the 10, however, it wins and the 10 is captured. The players have only a partial view on the opponent's pieces: seeing their position but not their type. The complete rules (*21*) are defined by the International Stratego Federation.



Replicator dynamics: $\frac{d}{dt}\pi_\tau^i(a^i) = \pi_\tau^i(a^i)\left[Q_{\pi_\tau}^i(a^i) - \sum_{b^i}\pi_\tau^i(b^i)Q_{\pi_\tau}^i(b^i)\right]$

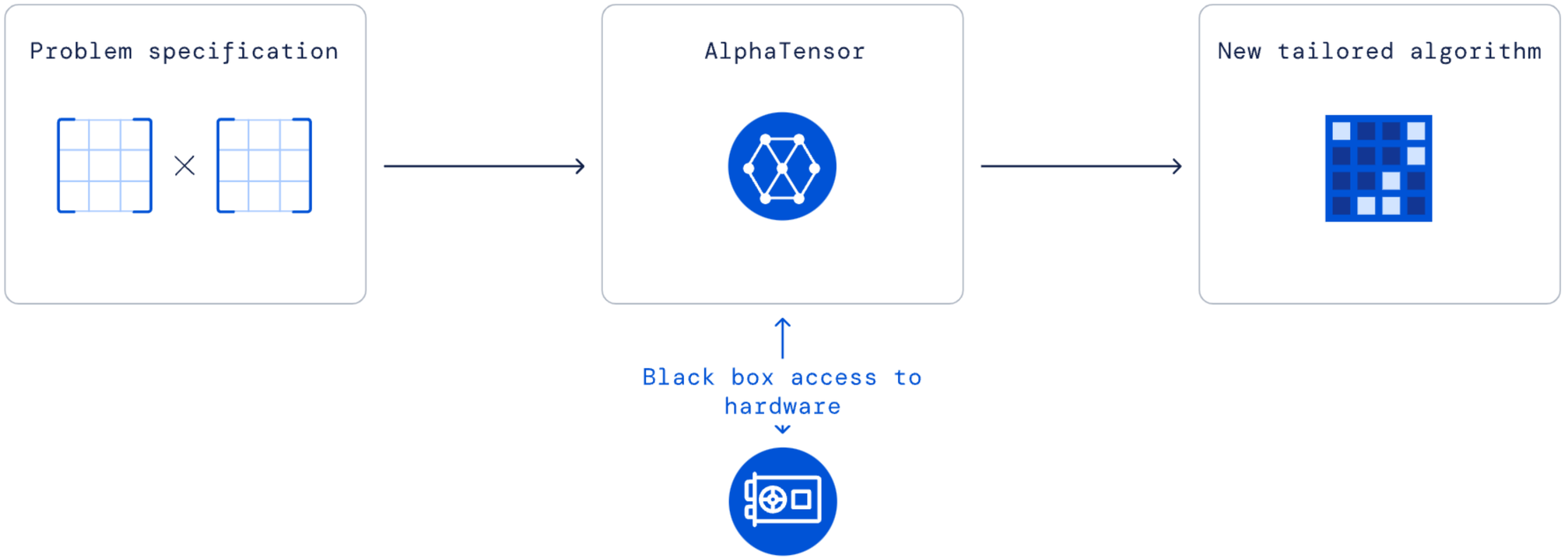
Reward transformation: $r^i(\pi^i, \pi^{-i}, a^i, a^{-i}) = r^i(a^i, a^{-i}) - \eta\log\left(\frac{\pi^i(a^i)}{\pi_{\mathrm{RB}}^i(a^i)}\right) + \eta\log\left(\frac{\pi^{-i}(a^{-i})}{\pi_{\mathrm{RB}}^{-i}(a^{-i})}\right)$

(b) An overview of the DeepNash approach. DeepNash is an autonomous agent that learns to play the imperfect information game Stratego (A). It learns a policy represented by a deep neural network (B) through self-play from scratch (C) in order to converge to a Nash equilibrium (D).

Figure 1: The Stratego game (a) and an overview of the *DeepNash* approach (b).

Examples: Discovering Novel Algorithms

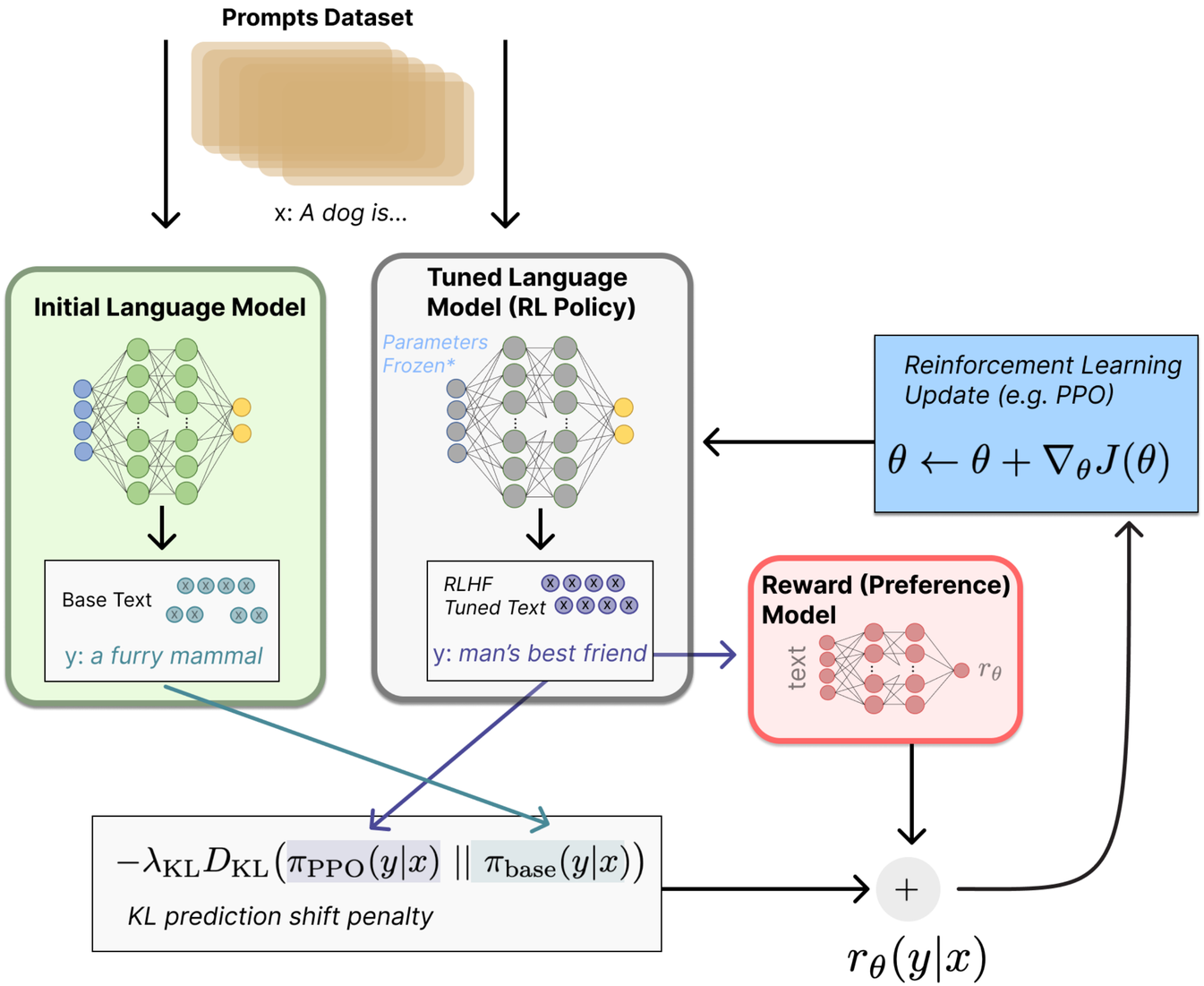
https://deepmind.google/discover/blog/discovering-novel-algorithms-with-alphatensor/



AlphaTensor with an objective corresponding to the runtime of the algorithm. When a correct matrix multiplication algorithm is discovered, it's benchmarked on the target hardware, which is then fed back to AlphaTensor, in order to learn more efficient algorithms on the target hardware.

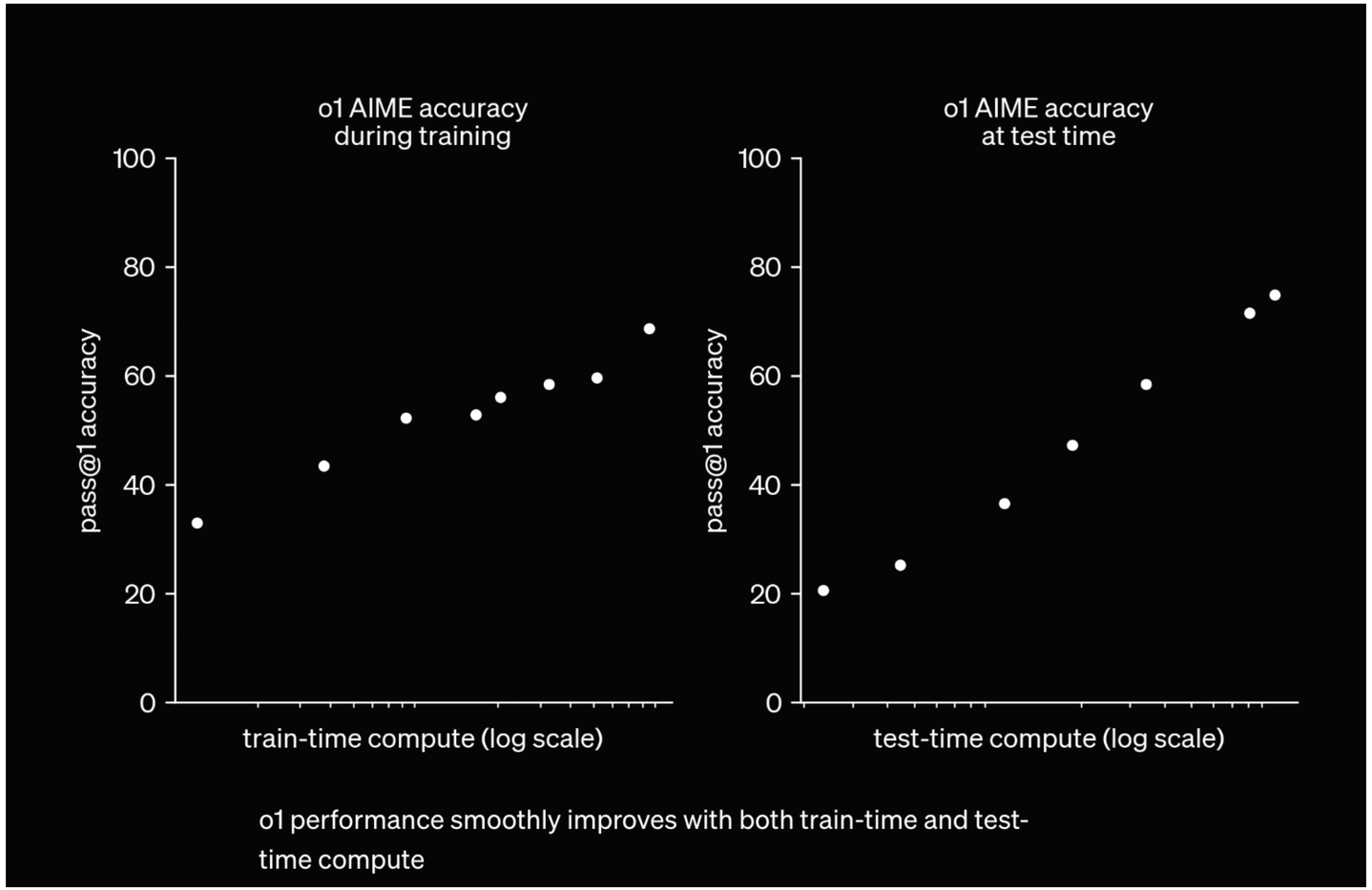
Examples: Alignment

https://huggingface.co/blog/rlhf

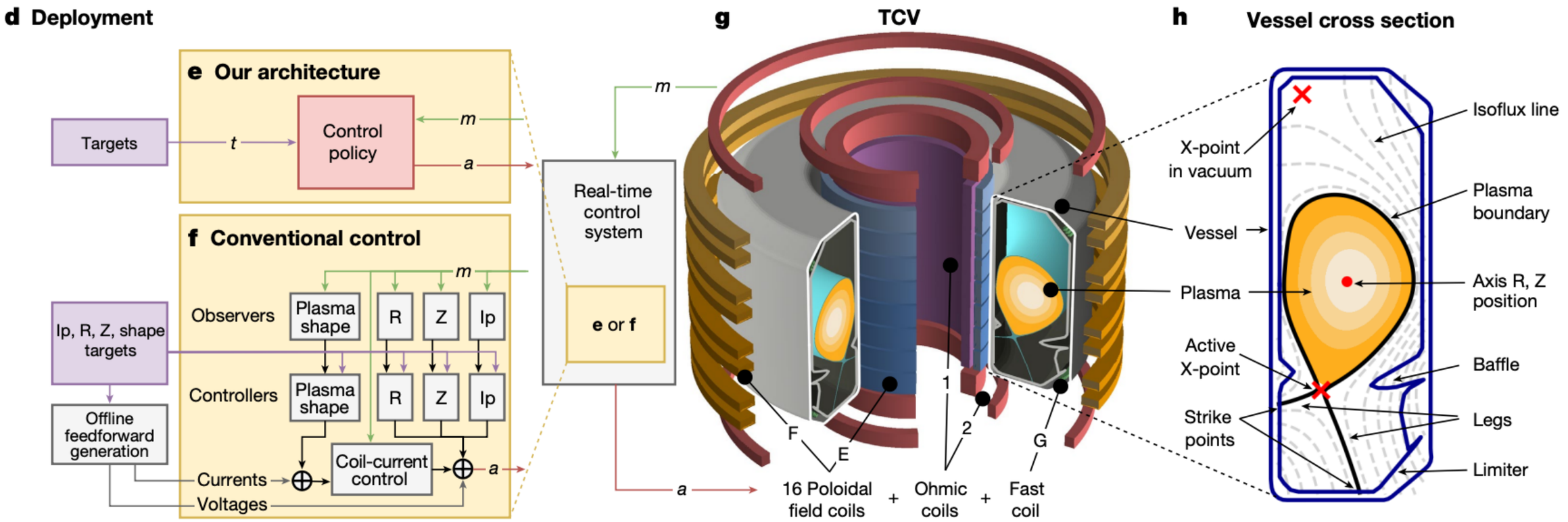


Examples: Reasoning

https://openai.com/index/learning-to-reason-with-llms/



Examples: Magnetic Control



https://www.nature.com/articles/s41586-021-04301-9

Examples: Trading

Examples: Trading

Reward

- A reward R_t is a scalar feedback signal
- Indicates how well agent is doing at step t
- The agent's job is to maximise cumulative reward

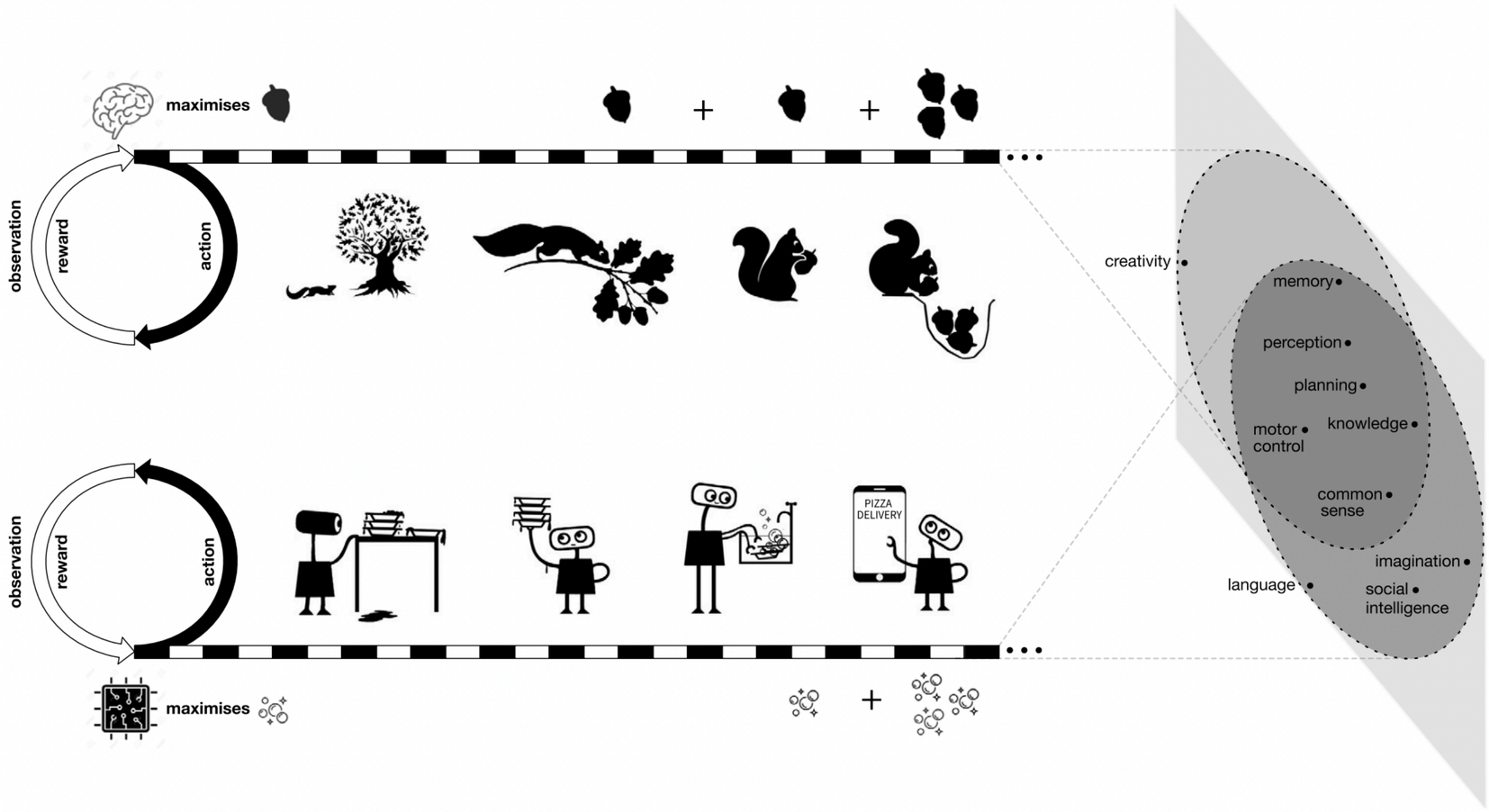
Reward hypothesis (Richard Sutton):

All goals can be described by the maximisation
of expected cumulative reward

Reward is enough

Reward

"Reward is enough to drive behaviour that exhibits abilities studied in natural and artificial intelligence, including knowledge, learning, perception, social intelligence, language, generalisation and imitation."

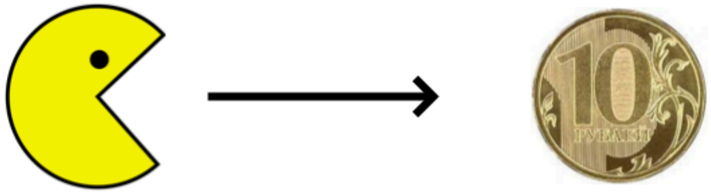


Reward Design

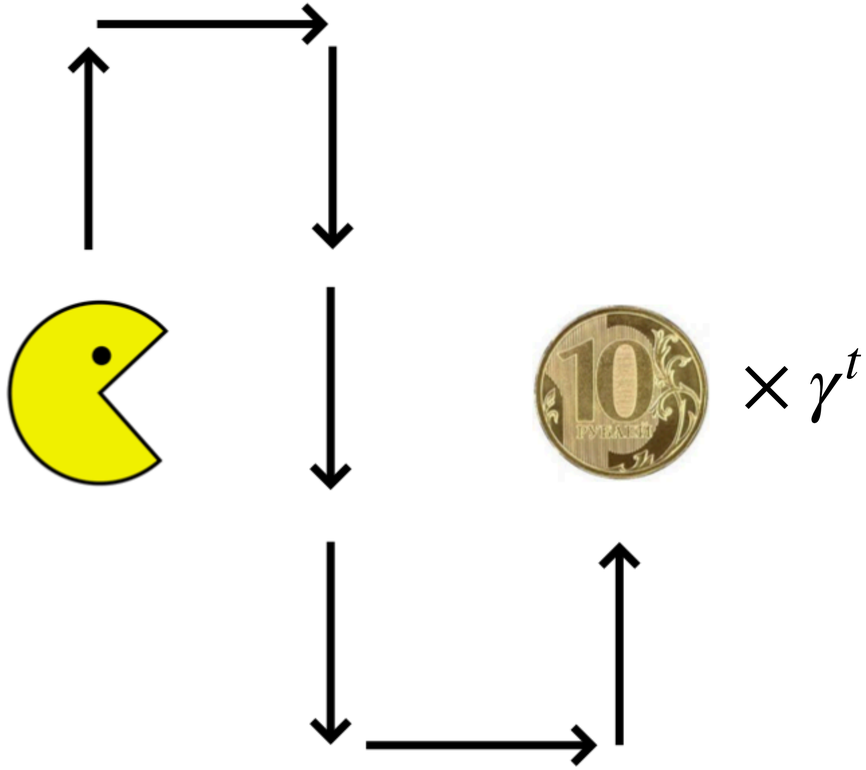
... is a tricky task

Discount Factor

$\gamma \in [0,1]$ is a discount factor



Source



Source

Why Discount?

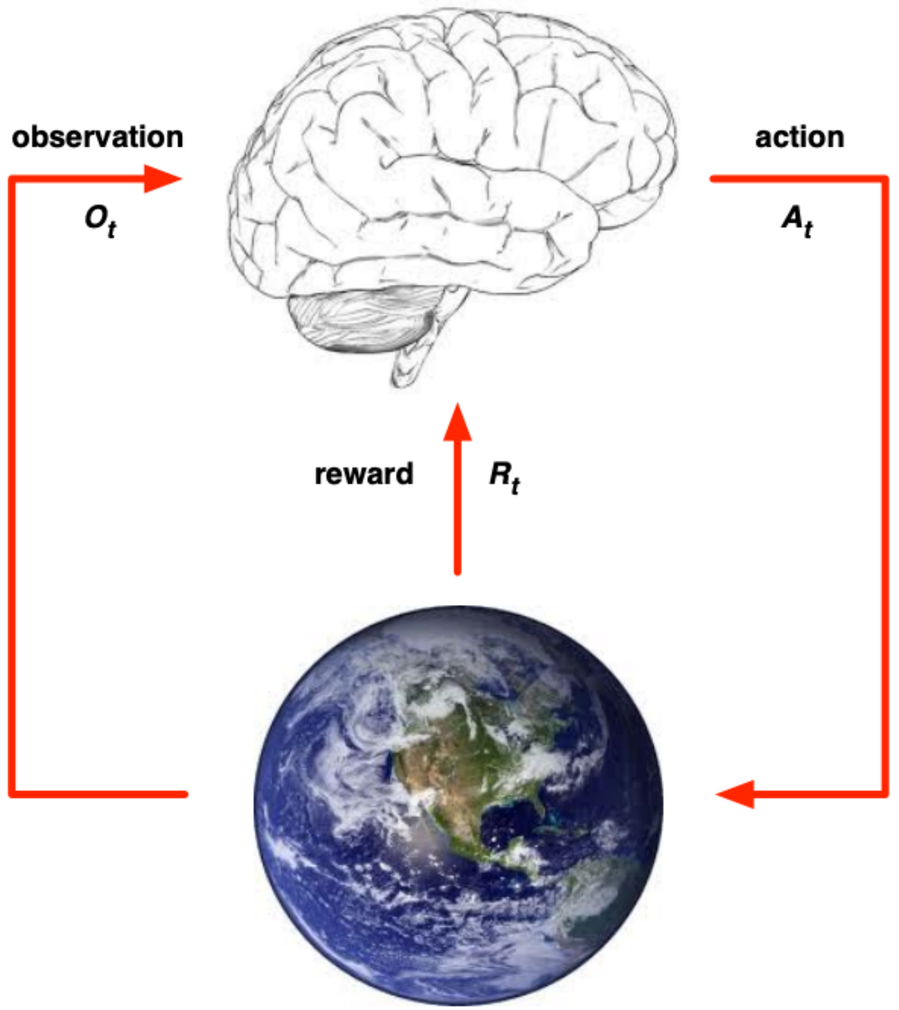
$\gamma \in [0,1]$ is a discount factor

- Avoids infinite returns: $\sum_{t=0}^{\infty} r$
- Uncertainty about the future may not be fully represented;
- Immediate rewards may earn more interest than delayed rewards due to volatility;
- Animal/human behaviour shows preference for immediate reward;
- It is sometimes possible to use undiscounted reward, e.g. if all sequences terminate, i.e. $T < +\infty$

Agent and Environment

At each step t the agent being at O_t:

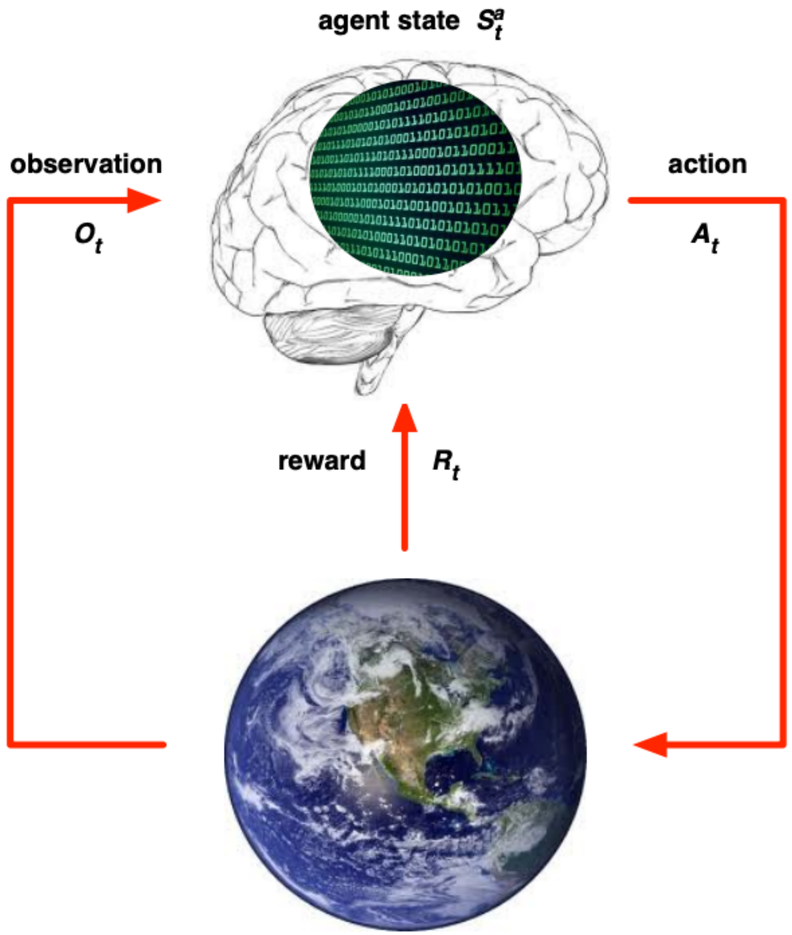
- Executes action A_t
- Receives scalar reward R_t
- Receives next observation O_{t+1}



Source

States

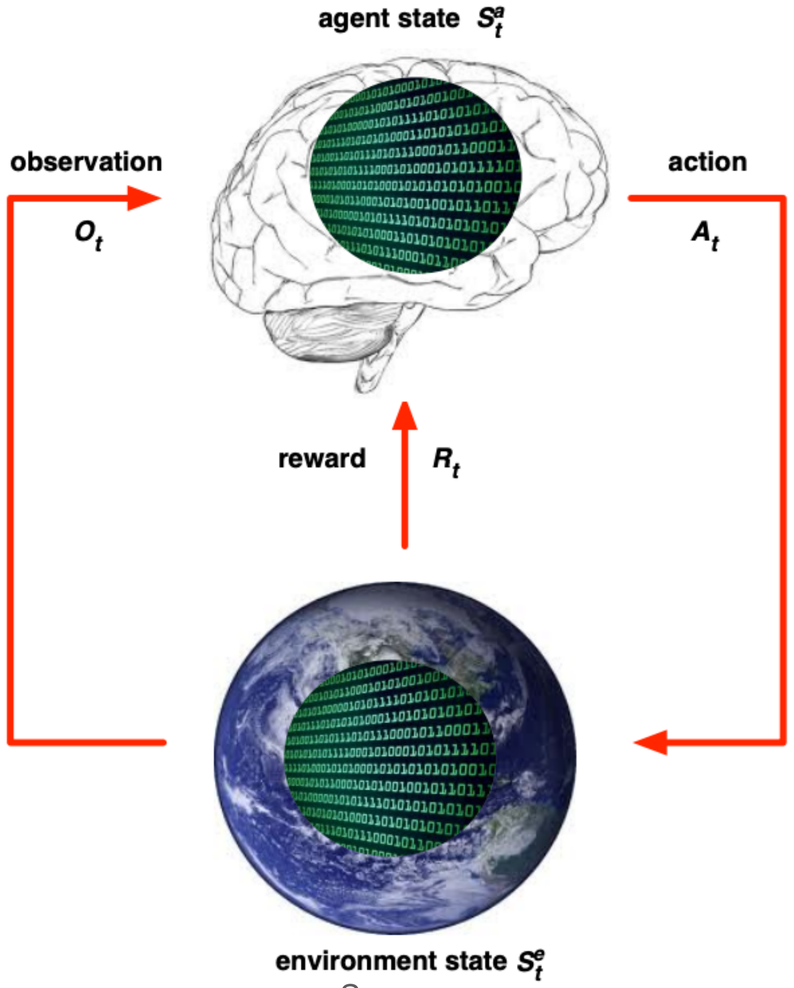
The environment state S_t^e is the environment's private representation.
The agent state S_t^a is the agent's internal representation.



Source

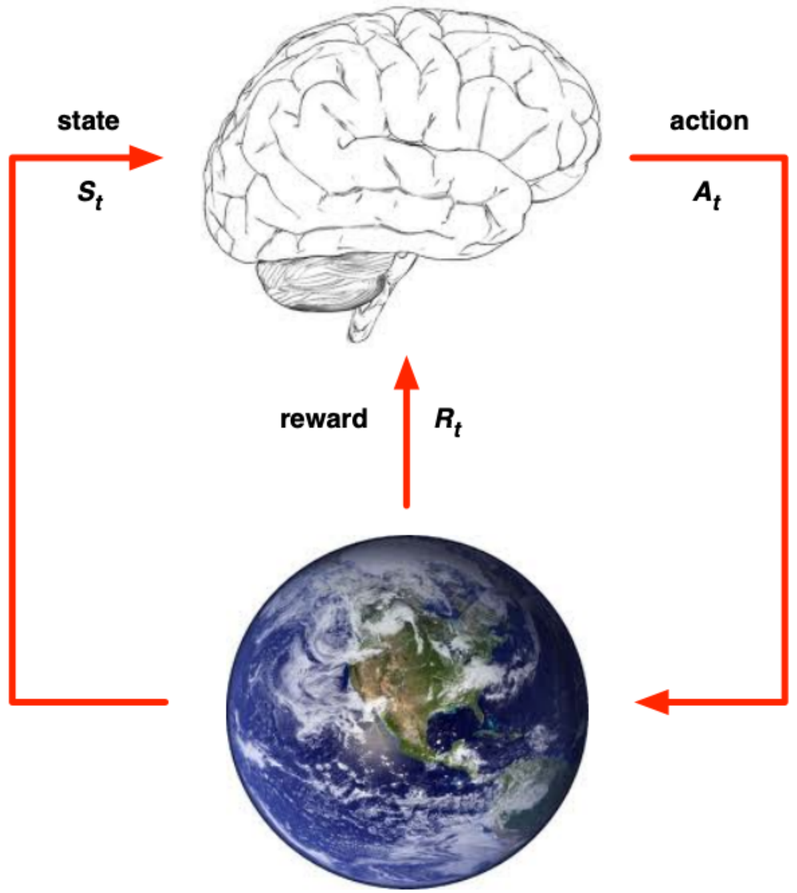
Observability

Partially observable: $S_t^e \neq S_t^a$



Source

Fully observable: $O_t = S_t^e = S_t^a$



Source

Policy

- A policy fully defines the behaviour of an agent
- It's a map from state to action, e.g.
- Deterministic policy: $a = \pi(s)$
- Stochastic policy: $\pi(a \mid s) = \mathbb{P}[A_t = a \mid S_t = s]$ (e.g. $a \sim \mathcal{N}(0,1)$)

Objective

$$\mathbb{E}_{\pi}[\sum_{t=0}^{T} \gamma^t R_t] \to \max_{\pi}$$

Plan

1. Intro + MDP + Dynamic Programming (HW1)
2. Model-free RL for tabular environments (HW2)
3. Intro to deep RL: DQN, RAINBOW and beyond (HW3)
4. Policy-based algorithms: REINFORCE, Actor-critic framework
5. Advanced policy-based algorithms: PPO (HW4)
6. Continuous control: DDPG, SAC, TQC
7. Offline RL (HW5)
8. Model-based RL: AlphaZero and beyond
9. Multi-armed bandits
10. RL in a context of LLM

Markov Decision Process

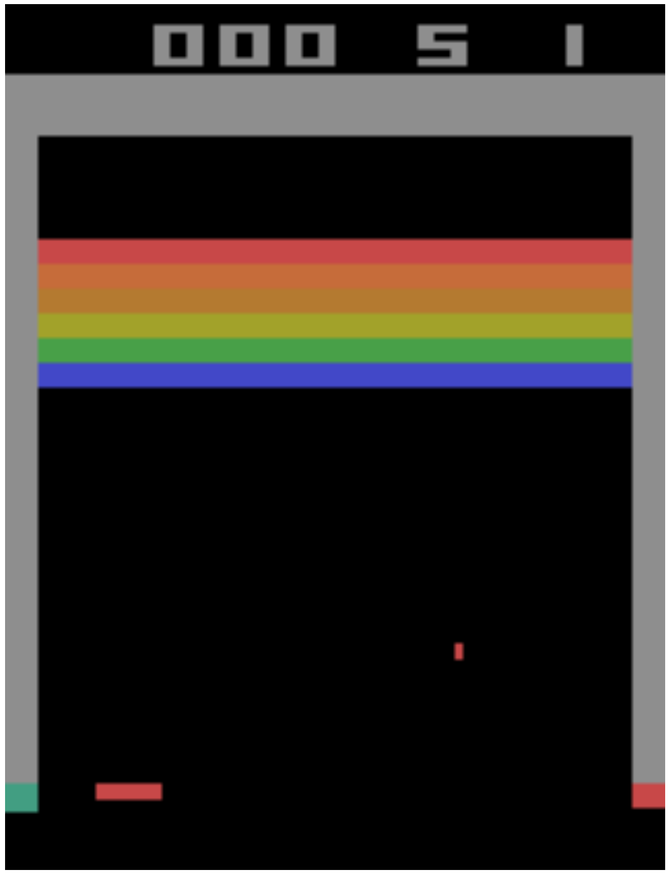
Markov Decision Process

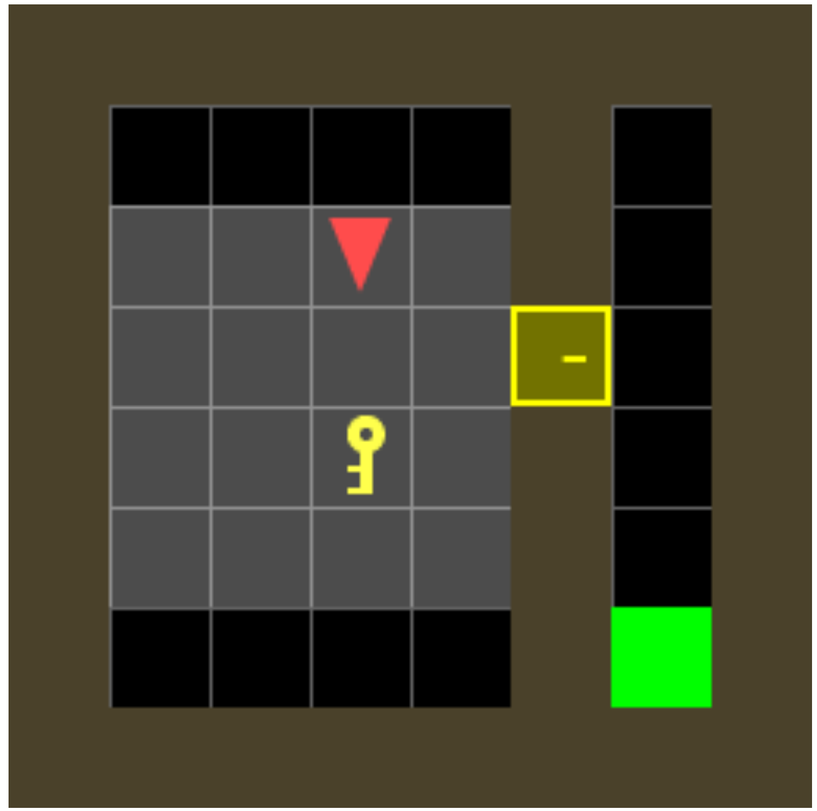
1. $\mathcal{A}$ is an action space
2. $\mathcal{S}$ is a state space
3. Initial state distribution: $s_0 \sim p_0$
4. $p(r, s' \mid s, a) = \mathbb{P}(R_t = r, S_{t+1} = s' \mid S_t = s, A_t = a)$ is the joint probability of a reward r and next state s', given a state s and action a
5. Discount factor $\gamma \in [0,1]$

Markov property:
(the future is independent of the past given the present)

$$p(R_t, S_{t+1} \mid S_t, A_t, R_{t-1}, S_{t-1}, A_{t-1}, \ldots) = p(R_t, S_{t+1} \mid S_t, A_t)$$

Non-markovian Environments

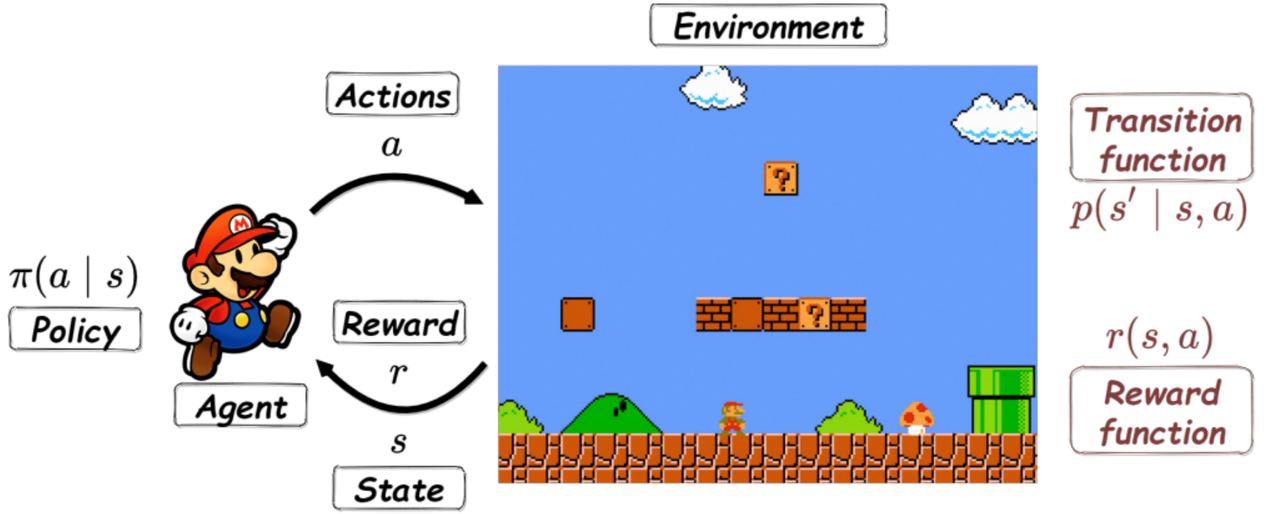




Equivalent Definition

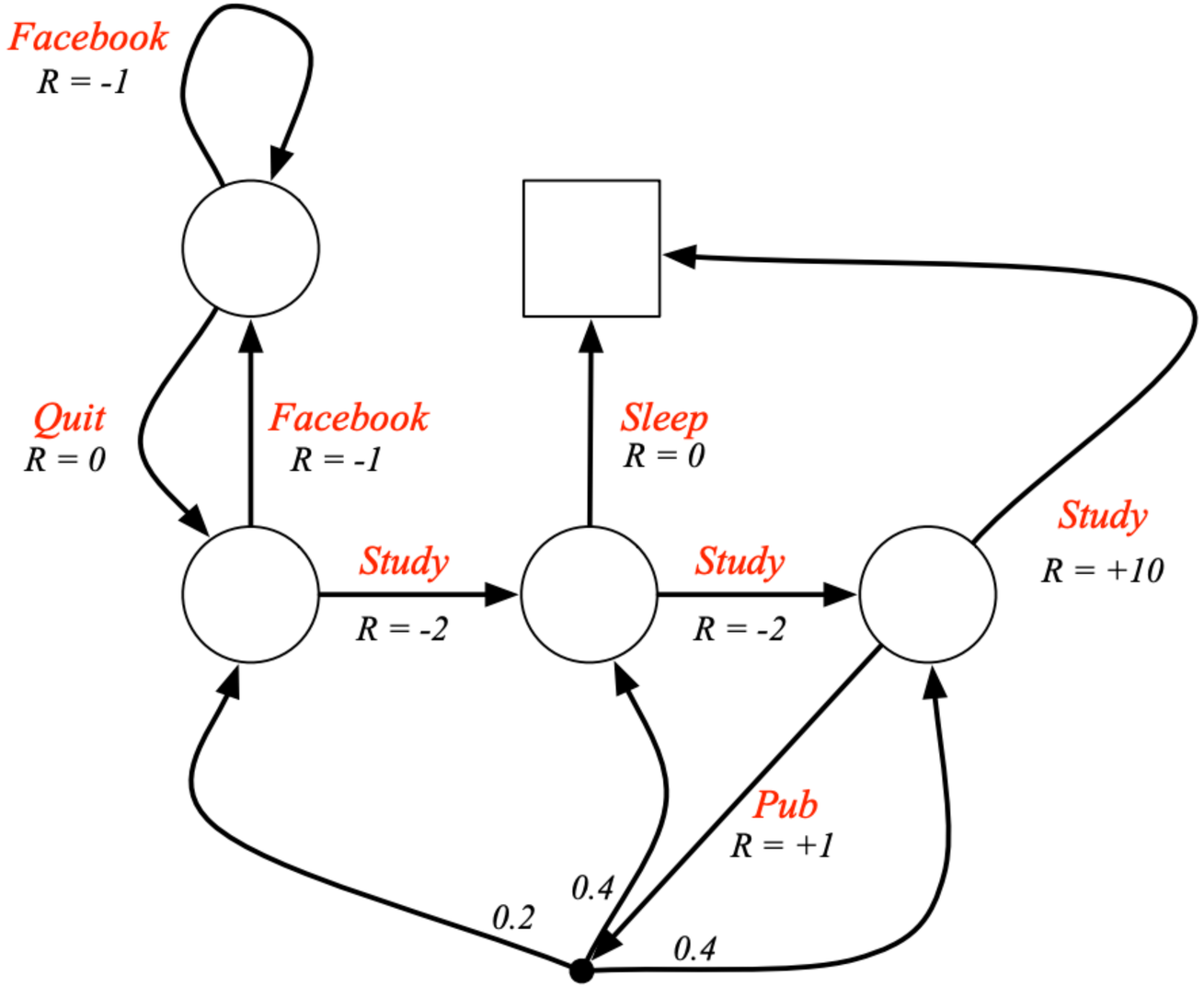
MDP is a 4-tuple $(\mathcal{S}, \mathcal{A}, p, r)$:

1. $\mathcal{A}$ is an action space
2. $\mathcal{S}$ is a state space
3. $p(s' \mid s, a) = \mathbb{P}(S_{t+1} = s' \mid S_t = s, A_t = a)$
   is a state-transition function
4. $r(s, a) \in \mathbb{R}$ is a reward function



Source

Example: Student MDP

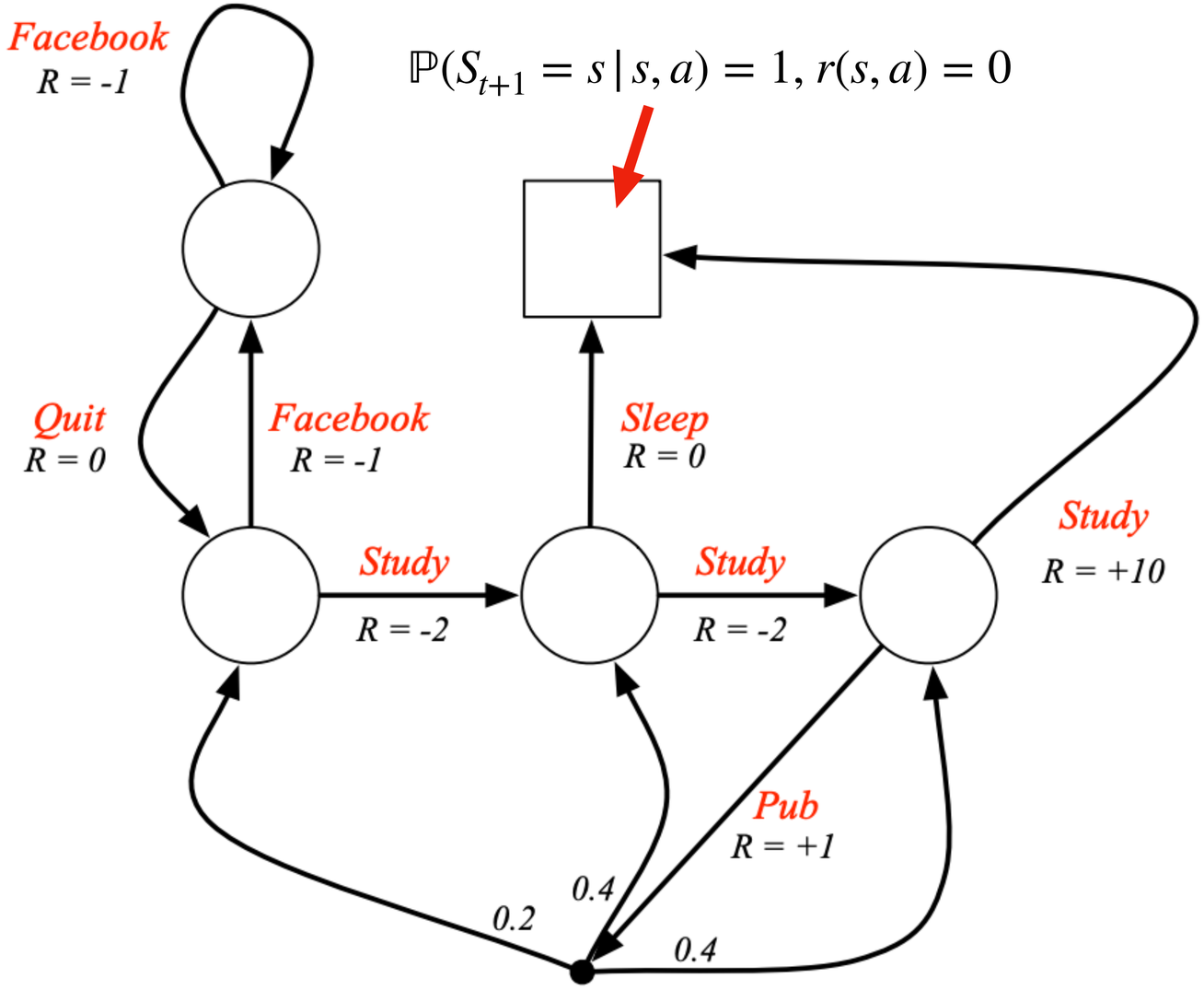


https://www.davidsilver.uk/wp-content/uploads/2020/03/MDP.pdf

Example: Student MDP

Terminal state:

$$\mathbb{P}(S_{t+1} = s \mid s, a) = 1, r(s, a) = 0$$



https://www.davidsilver.uk/wp-content/uploads/2020/03/MDP.pdf

Tabular MDP:

$$|\mathcal{A}| < \infty, |\mathcal{S}| < \infty$$

Returns and Episodes

Let T is a final time step. If $T < \infty$ then environment is called *episodic*.

Cumulative reward is called a return or reward-to-go. Note that in general it is a random variable.

$$G_t = R_t + \gamma R_{t+1} + \gamma^2 R_{t+2} + \ldots = \sum_{k=t}^{T} \gamma^{k-t} R_k$$

Episode length

Immediate reward

Discount factor

Returns and Episodes

Let T is a final time step. If $T < \infty$ then environment is called *episodic*.

Cumulative reward is called a return or reward-to-go. Note that in general it is a random variable.

$$G_t = R_t + \gamma R_{t+1} + \gamma^2 R_{t+2} + \ldots = \sum_{k=t}^{T} \gamma^{k-t} R_k$$

Episode length

Immediate reward

Discount factor

Important note: $G_t = R_t + \gamma(R_{t+1} + \gamma R_{t+2} + \ldots) = R_t + \gamma G_{t+1}$

Objective Function

- Agent interacts with the environment following some policy $\pi(a \mid s)$.
- Policy stochasticity: $a \sim \pi(\,. \mid s)$
- Environment stochasticity: $s' \sim p(\,. \mid s, a)$
- A trajectory: $\tau = (s_0, a_0, s_1, \ldots, s_{T-1}, a_{T-1}, s_T)$

$$J(\pi) = \mathbb{E}_{\pi}[G_0] = \mathbb{E}_{p(\tau|\pi)}[G_0] =$$

$$= \mathbb{E}_{s_0 \sim p_0}[\mathbb{E}_{a_0 \sim \pi(.|s_0)}[\mathbb{E}_{s_1 \sim p(.|s_0,a_0)}[r_0 + \gamma[\ldots]]]] =$$

$$= \mathbb{E}_{s_0}[\mathbb{E}_{a_0}[\mathbb{E}_{s_1}[r_0 + \gamma[\ldots]]]] \to \max_{\pi}$$

State-value Function

Estimate how good it is for an agent to be in a given state s and follow a policy π:

$$V^{\pi}(s) = \mathbb{E}_{\pi}[G_t \,|\, S_t = s] = \mathbb{E}_{\pi}[\sum_{k=0}^{T} \gamma^k R_{t+k} \,|\, S_t = s]$$

- If s is a terminal state then $V^{\pi}(s) = 0$.
- $J(\pi) = \mathbb{E}_{s_0} V^{\pi}(s_0)$

Action-value Function

Estimate how good it is for the agent to take an action a being in a given state s and follow a policy π:

$$Q^{\pi}(s,a) = \mathbb{E}_{\pi}[G_t \,|\, S_t = s, A_t = a] = \mathbb{E}_{\pi}[\sum_{k=0}^{T} \gamma^k R_{t+k} \,|\, S_t = s, A_t = a]$$

- If s is a terminal state then $Q^{\pi}(s,a) = 0 \; \forall a \in \mathscr{A}$.
- $V^{\pi}(s) = \mathbb{E}_{a \sim \pi(.|s)} Q(s,a)$
- Advantage: $A^{\pi}(s,a) = Q^{\pi}(s,a) - V^{\pi}(s)$

Bellman Equations

State-value Function

Law of iterated expectation

$$V^{\pi}(s) = \mathbb{E}_{\pi}[G_t \,|\, S_t = s] = \mathbb{E}_{\pi}[R_t + \gamma G_{t+1} \,|\, S_t = s] =$$

$$= \sum_{a} \pi(a \,|\, s) \sum_{s'} \mathbb{P}(S_{t+1} = s' \,|\, S_t = s, A_t = a, S_{t-1} = s'', A_{t-1} = a'', \ldots) \Big[r + \gamma \mathbb{E}_{\pi}[G_{t+1} \,|\, S_{t+1} = s'] \Big] =$$

Definition of state transition function

$$= \sum_{a} \pi(a \,|\, s) \sum_{s'} \mathbb{P}(S_{t+1} = s' \,|\, S_t = s, A_t = a)[\ldots] =$$

Markov property

Definition of $V^{\pi}(s')$

$$= \sum_{a} \pi(a \,|\, s) \sum_{s'} p(s' \,|\, s, a) \Big[r + \gamma \mathbb{E}_{\pi}[G_{t+1} \,|\, S_{t+1} = s'] \Big]$$

$$= \sum_{a} \pi(a \,|\, s) \sum_{s'} p(s' \,|\, s, a) \Big[r + \gamma V^{\pi}(s') \Big] = \mathbb{E}_{a}[r(s, a) + \gamma \mathbb{E}_{s'} V^{\pi}(s')]$$

Policy stochasticity

Environment stochasticity

Action-value Function

$$Q^{\pi}(s,a) = \mathbb{E}_{\pi}[G_t \,|\, S_t = s, A_t = a] = \mathbb{E}_{\pi}[R_t + \gamma G_{t+1} \,|\, S_t = s, A_t = a] =$$

$$= \sum_{s'} p(s' \,|\, s, a) \Big[r + \gamma \mathbb{E}_{\pi}[G_{t+1} \,|\, S_{t+1} = s'] \Big] = \sum_{s'} p(s' \,|\, s, a) \Big[r + \gamma V^{\pi}(s') \Big] =$$

$$= r(s,a) + \gamma \mathbb{E}_{s'} V^{\pi}(s')$$

Relationship between V^π and Q^π

$$V^\pi(s) = \sum_a \pi(a \mid s) \sum_{s'} p(s' \mid s, a) \left[r + \gamma V^\pi(s') \right]$$

$$Q^\pi(s, a) = \sum_{s'} p(s' \mid s, a) \left[r + \gamma V^\pi(s') \right]$$

Relationship between V^{π} and Q^{π}

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

$$Q^{\pi}(s, a) = \sum_{s'} p(s' \mid s, a) \left[r + \gamma V^{\pi}(s') \right]$$

$$V_{\pi}(s) = \sum_{a} \pi(a \mid s) Q_{\pi}(s, a) = \mathbb{E}_{a \sim \pi(. \mid s)} Q_{\pi}(s, a)$$

Relationship between V^{π} and Q^{π}

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

$$Q^{\pi}(s, a) = \sum_{s'} p(s' \mid s, a) \left[r + \gamma V^{\pi}(s') \right]$$

$$V_{\pi}(s) = \sum_{a} \pi(a \mid s) Q_{\pi}(s, a) = \mathbb{E}_{a \sim \pi(. \mid s)} Q_{\pi}(s, a)$$

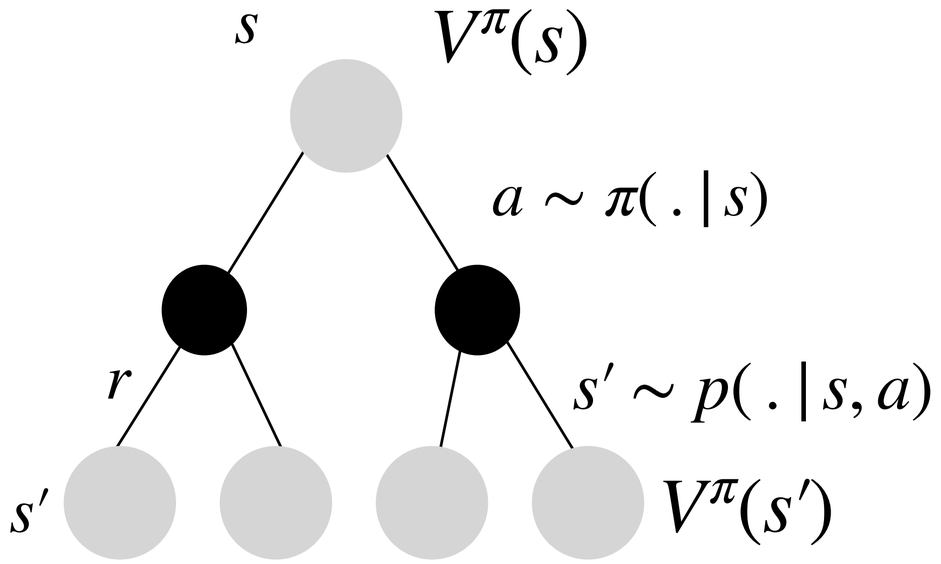
$$Q^{\pi}(s, a) = \sum_{s'} p(s' \mid s, a) \left[r + \gamma \sum_{a'} \pi(a' \mid s') Q^{\pi}(s', a') \right]$$

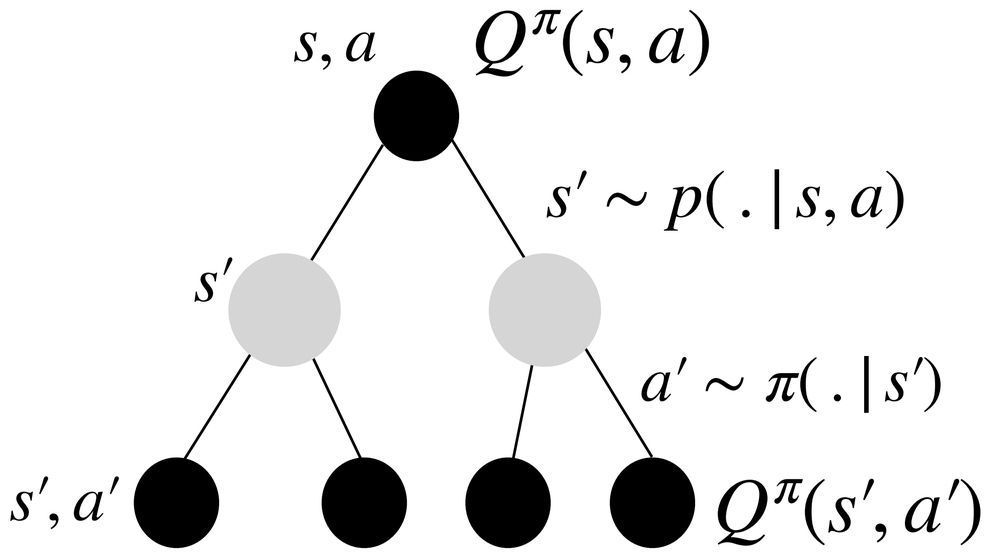
Bellman Expectations Equations

- $V^{\pi}(s) = \sum_{a} \pi(a \mid s) \sum_{s'} p(s' \mid s, a) \left[r + \gamma V^{\pi}(s') \right] = \mathbb{E}_{a}[r(s, a) + \gamma \mathbb{E}_{s'} V^{\pi}(s')]$
- $Q^{\pi}(s, a) = \sum_{s'} p(s' \mid s, a) \left[r + \gamma \sum_{a'} \pi(a' \mid s') Q^{\pi}(s', a') \right] = r(s, a) + \gamma \mathbb{E}_{s'} \mathbb{E}_{a'} Q^{\pi}(s', a')$

Bellman Expectations Equations

- $V^{\pi}(s) = \sum_{a} \pi(a \mid s) \sum_{s'} p(s' \mid s, a) \left[r + \gamma V^{\pi}(s') \right] = \mathbb{E}_{a}[r(s, a) + \gamma \mathbb{E}_{s'} V^{\pi}(s')]$
- $Q^{\pi}(s, a) = \sum_{s'} p(s' \mid s, a) \left[r + \gamma \sum_{a'} \pi(a' \mid s') Q^{\pi}(s', a') \right] = r(s, a) + \gamma \mathbb{E}_{s'} \mathbb{E}_{a'} Q^{\pi}(s', a')$





Optimal Value Function

Recall that our goal is to find a policy that achieves maximum expected cumulative discounted reward.

Define optimal value functions:

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

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

- The optimal value function specifies the best possible performance in the MDP.
- An MDP is “solved” when we know the optimal value function.

Optimal Policy

Define a partial order on a set of possible policies:
$\pi \geq \pi' \iff V^{\pi}(s) \geq V^{\pi'}(s) \, \forall s \in \mathcal{S}$

Theorem: For any MDP

- There exists an optimal policy π^* that is better than or equal to all other policies, $\pi^* \geq \pi \, \forall \pi$
- All optimal policies achieve the optimal value functions

Finding an Optimal Policy

- An optimal policy can be found: $\pi^*(a \mid s) = \mathbb{I}[a = argmax_{a'} Q^*(s, a')]$
- There is always a deterministic optimal policy for any MDP

Bellman Optimality Equations

$$V^*(s) = \mathbb{E}_{a\sim\pi^*} Q^{\pi^*}(s,a) \quad \longrightarrow \quad V^*(s) = \max_a Q^*(s,a)$$

$$V^*(s) = \max_a \mathbb{E}_{\pi^*}[G_t \,|\, S_t = s, A_t = a] = \max_a \mathbb{E}_{\pi^*}[R_t + \gamma G_{t+1} \,|\, S_t = s, A_t = a] =$$

$$= \max_a \mathbb{E}_{\pi^*}[R_t + \gamma V^*(S_{t+1}) \,|\, S_t = s, A_t = a] = \max_a \sum_r p(s' \,|\, s, a)[r + \gamma V^*(s')] =$$

$$= \max_a [r(s,a) + \gamma \mathbb{E}_{s'} V^*(s')]$$

Bellman Optimality Equations

$$Q^{\pi}(s,a) = \sum_{s'} p(s' \mid s,a)\left[r + \gamma V^{\pi}(s')\right]$$

$$Q^{*}(s,a) = \mathbb{E}[R_t + \gamma V^{*}(S_{t+1}) \mid S_t = s, A_t = a] = \sum_{s'} p(s' \mid s,a)\left[r + \gamma \max_{a'} Q^{*}(s',a')\right] =$$

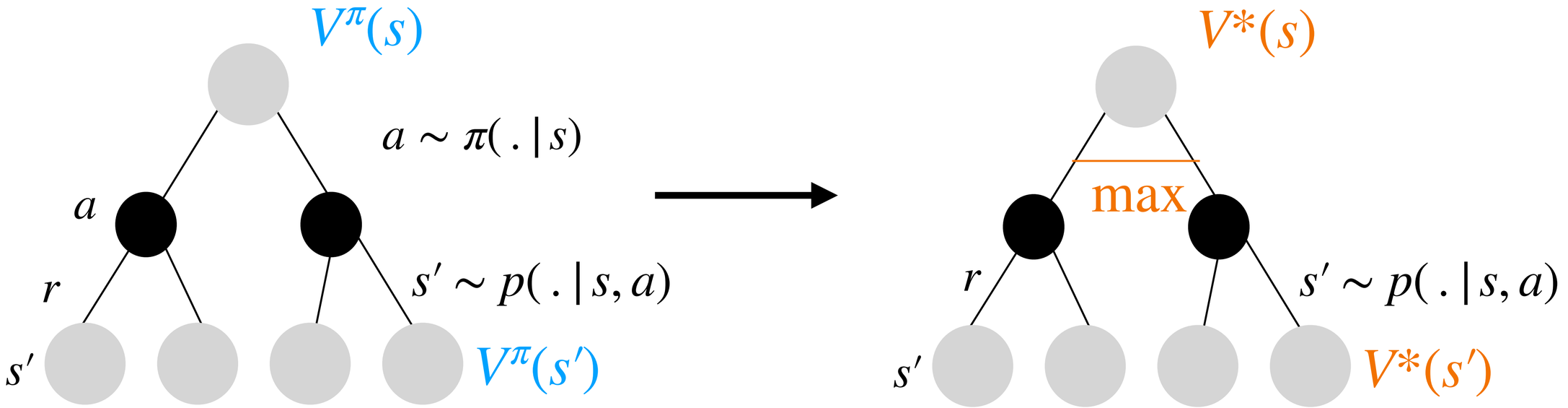
$$= r(s,a) + \gamma \mathbb{E}_{s'} \max_{a'} Q^{*}(s',a')$$

Bellman Optimality Equations

$$V^*(s) = \max_a [r(s,a) + \gamma \sum_{s'} p(s' \mid s,a) V^*(s')] = \max_a [r(s,a) + \gamma \mathbb{E}_{s'} V^*(s')]$$

$$Q^*(s,a) = r(s,a) + \gamma \sum_{s'} p(s' \mid s,a) \max_{a'} Q^*(s',a') = r(s,a) + \gamma \mathbb{E}_{s'} \max_{a'} Q^*(s',a')$$

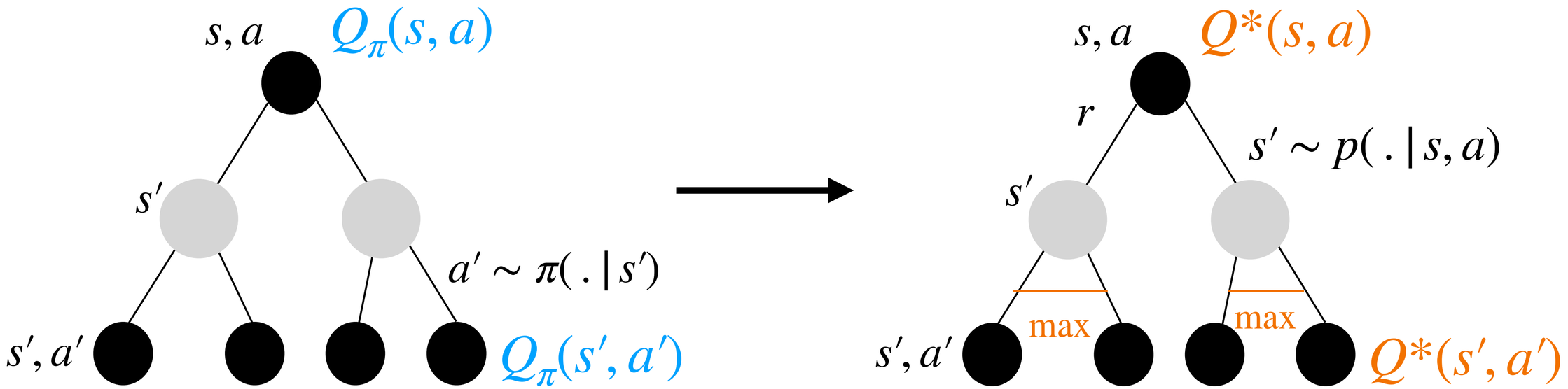
Intuition



$$V^{\pi}(s) = \mathbb{E}_{a}[r(s,a) + \gamma \mathbb{E}_{s'} V^{\pi}(s')]$$

$$V^{*}(s) = \max_{a}[r(s,a) + \gamma \mathbb{E}_{s'} V^{*}(s')]$$

Intuition



$$Q^\pi(s, a) = r(s, a) + \gamma \mathbb{E}_{a'} \mathbb{E}_{s'} Q^\pi(s', a')$$

$$Q^*(s, a) = r(s, a) + \gamma \mathbb{E}_{s'} \max_{a'} Q^*(s', a')$$

Recover Optimal Policy

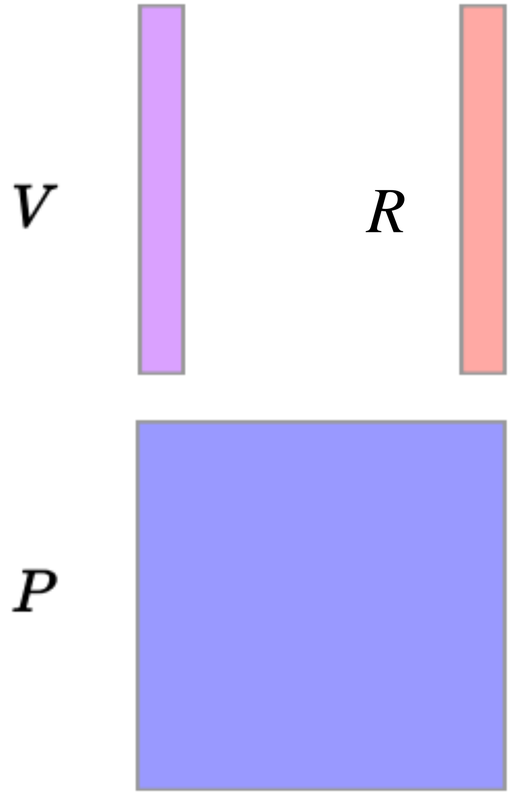
1. If Q^* is known: $\pi(s) = argmax_a Q^*(s, a)$
2. If V^* is known: $\pi(s) = argmax_a \sum_{s'} p(s' \mid s, a)[r + \gamma V^*(s')]$

Linear Equations

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

is a system of $|S|$ linear equations:

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



Source

Linear Equations

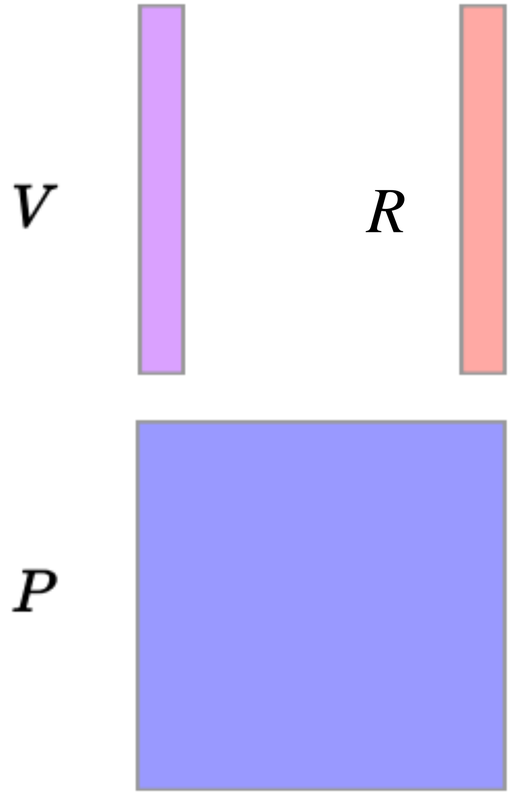
$$V^{\pi}(s) = \sum_{a} \pi(a \mid s) \sum_{s'} p(s' \mid s, a) \left[r + \gamma V^{\pi}(s') \right]$$

is a system of $|S|$ linear equations:

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

General case: $\mathcal{O}(|S|^3)$

Maybe we can try iterative solution methods?



Source

Dynamic Programming

Problems in RL

1. Estimating V^{π} or Q^{π} is called policy evaluation or prediction

2. Estimating V^{*} or Q^{*} is called control because they can be used for policy optimisation

Bellman Operators

Bellman expectation operator for V: $[\mathscr{B}^{\pi}V](s) = \mathbb{E}_{a\sim\pi(.|s)}[r(s,a) + \gamma\mathbb{E}_{s'\sim p(.|s,a)}V(s')]$

Bellman expectation operator for Q: $[\mathscr{B}^{\pi}Q](s,a) = r(s,a) + \gamma\mathbb{E}_{s'\sim\pi(.|s,a)}[Q(s',a')]$

Bellman optimality operator for V: $[\mathscr{B}^{*}V](s) = \max_{a}[r(s,a) + \gamma\mathbb{E}_{s'\sim p(.|s,a)}V(s')]$

Bellman optimality operator for Q: $[\mathscr{B}^{*}Q](s,a) = r(s,a) + \gamma\mathbb{E}_{s'\sim p(.|s,a)}\max_{a'}Q(s',a')$

Bellman Operators

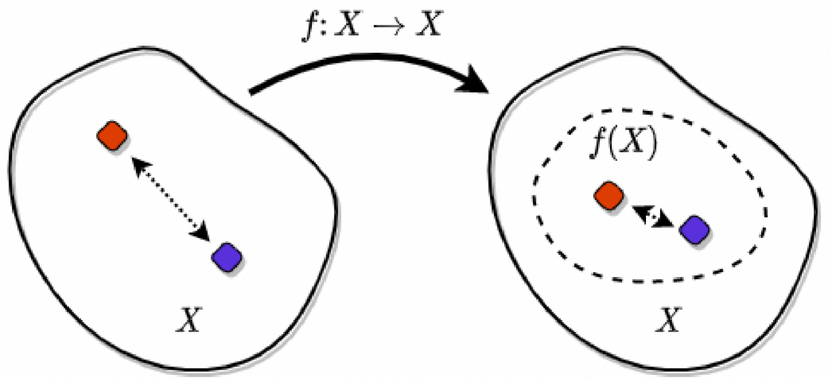
Bellman expectation operator for V: $[\mathscr{B}^{\pi}V](s) = \mathbb{E}_{a\sim\pi(.|s)}[r(s,a) + \gamma\mathbb{E}_{s'\sim p(.|s,a)}V(s')]$

Bellman expectation operator for Q: $[\mathscr{B}^{\pi}Q](s,a) = r(s,a) + \gamma\mathbb{E}_{s'\sim\pi(.|s,a)}[Q(s',a')]$

Bellman optimality operator for V: $[\mathscr{B}^{*}V](s) = \max_{a}[r(s,a) + \gamma\mathbb{E}_{s'\sim p(.|s,a)}V(s')]$

Bellman optimality operator for Q: $[\mathscr{B}^{*}Q](s,a) = r(s,a) + \gamma\mathbb{E}_{s'\sim p(.|s,a)}\max_{a'}Q(s',a')$

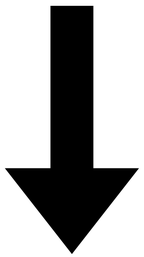
All these operators are contractions with coefficient γ w.r.t. $||.||_{\infty}$, so by Banach fixed point theorem there exists a unique fixed point.



Policy Evaluation

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

Linear equations



$$V_{k+1}(s) = [\mathscr{B}^{\pi} V_k](s) = \sum_{a} \pi(a \mid s) \sum_{s'} p(s' \mid s, a) \left[r + \gamma V_k(s') \right]$$

for fixed policy π

Simple iteration method

Policy Evaluation Algorithm

Input π, the policy to be evaluated
Initialize an array $V(s) = 0$, for all $s \in \mathcal{S}^+$
Repeat
    $\Delta \leftarrow 0$
    For each $s \in \mathcal{S}$:
        $v \leftarrow V(s)$
        $V(s) \leftarrow \sum_a \pi(a|s) \sum_{s',r} p(s', r|s, a)\left[r + \gamma V(s')\right]$
        $\Delta \leftarrow \max(\Delta, |v - V(s)|)$
until $\Delta < \theta$ (a small positive number)
Output $V \approx v_\pi$

Bellman expectation operator for $V^\pi(s)$

Policy Improvement

Consider deterministic policy: $a = \pi(s)$

Let us act greedily w.r.t. $Q^{\pi}(s, a)$:

$$\pi'(s) = argmax_a Q^{\pi}(s, a) = argmax_a \sum_{s'} p(s' \mid s, a)[r + \gamma V^{\pi}(s')]$$

Policy Improvement

- $\pi'(s) = argmax_a Q^\pi(s,a)$
- $Q^\pi(s, \pi'(s)) = \max_a Q^\pi(s,a) \geq V^\pi(s) \; \forall s$
- π' is not worse than π:

$$V^\pi(s) \leq Q^\pi(s, \pi'(s)) = r(s, \pi'(s)) + \gamma \mathbb{E}_{s'} V^\pi(s') + \leq$$

$$\leq r(s, \pi'(s)) + \gamma \mathbb{E}_{s'} Q^\pi(s', \pi'(s')) \leq$$

$$\leq r(s, \pi'(s)) + \gamma r(s', \pi'(s')) + \gamma^2 \mathbb{E}_{s''} V^\pi(s'') \leq \ldots \leq V^{\pi'}(s)$$

Policy Improvement

- If improvements stop:

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

- Then the Bellman optimality equation has been satisfied:

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

- Therefore, $V^{\pi} = V^{*}$, so π is an optimal policy.

Policy Iteration

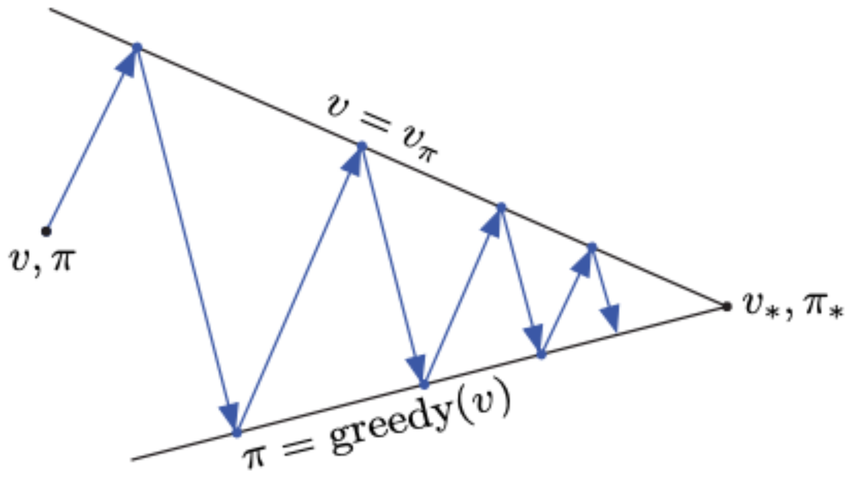
1. Initialization
   $V(s) \in \mathbb{R}$ and $\pi(s) \in \mathcal{A}(s)$ arbitrarily for all $s \in \mathcal{S}$

2. Policy Evaluation
   Repeat
      $\Delta \leftarrow 0$
      For each $s \in \mathcal{S}$:
         $v \leftarrow V(s)$
         $V(s) \leftarrow \sum_{s',r} p(s', r|s, \pi(s))\left[r + \gamma V(s')\right]$
         $\Delta \leftarrow \max(\Delta, |v - V(s)|)$
   until $\Delta < \theta$ (a small positive number)

   Bellman expectation operator for $V^{\pi}(s)$

3. Policy Improvement
   policy-stable $\leftarrow$ *true*
   For each $s \in \mathcal{S}$:
      old-action $\leftarrow \pi(s)$
      $\pi(s) \leftarrow \arg\max_a \sum_{s',r} p(s', r|s, a)\left[r + \gamma V(s')\right]$
      If *old-action* $\neq \pi(s)$, then *policy-stable* $\leftarrow$ *false*
   If *policy-stable*, then stop and return $V \approx v_*$ and $\pi \approx \pi_*$; else go to 2

   Bellman optimality operator for $V^{\pi}(s)$



$$\pi_0 \xrightarrow{\text{E}} v_{\pi_0} \xrightarrow{\text{I}} \pi_1 \xrightarrow{\text{E}} v_{\pi_1} \xrightarrow{\text{I}} \pi_2 \xrightarrow{\text{E}} \cdots \xrightarrow{\text{I}} \pi_* \xrightarrow{\text{E}} v_*$$

Source

Value Iteration

In fact, the policy evaluation step of policy iteration can be truncated in several ways without losing the convergence guarantees of policy iteration. One important special case is when policy evaluation is stopped after just one sweep (one update of each state). This algorithm is called value iteration.

$$V_{k+1}(s) = \max_a \sum_{s'} p(s' \mid s, a) \left[r + \gamma V_k(s') \right]$$

Value Iteration

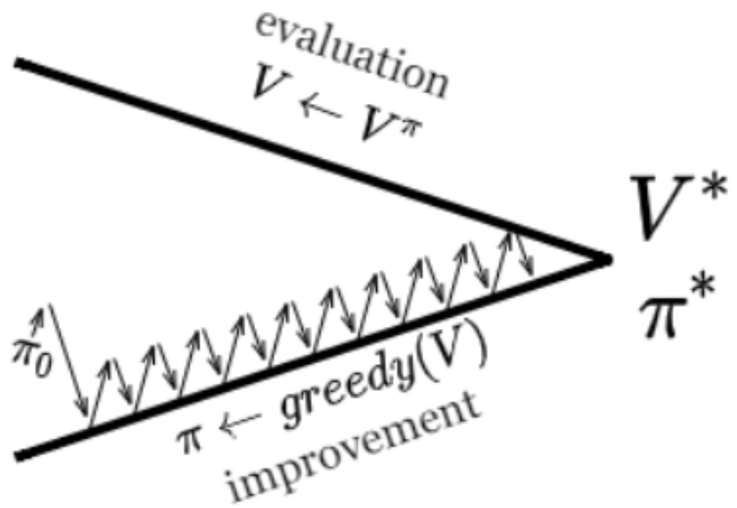
Initialize array V arbitrarily (e.g., $V(s) = 0$ for all $s \in \mathcal{S}^+$)

Repeat
  $\Delta \leftarrow 0$
  For each $s \in \mathcal{S}$:
    $v \leftarrow V(s)$
    $V(s) \leftarrow \max_a \sum_{s',r} p(s', r|s, a)\big[r + \gamma V(s')\big]$
    $\Delta \leftarrow \max(\Delta, |v - V(s)|)$
until $\Delta < \theta$ (a small positive number)

Output a deterministic policy, $\pi \approx \pi_*$, such that
  $\pi(s) = \operatorname{argmax}_a \sum_{s',r} p(s', r|s, a)\big[r + \gamma V(s')\big]$

Generalised Policy Iteration

- Perform N iterations of Policy Evaluation
- Perform Policy Improvement



- $N = 1$ - Value Iteration
- $N = \infty$ - Policy Iteration

Synchronous Dynamic Programming Algorithms

Problem	Bellman Equation	Algorithm	Complexity per iteration:
Prediction	Bellman Expectation Equation	Iterative Policy Evaluation	
Control	Bellman Expectation Equation + (Greedy) Policy Improvement	Policy Iteration	$O(\|A\|\|S\|^2 + \|S\|^3)$
Control	Bellman Optimality Equation	Value Iteration	$O(\|A\|\|S\|^2)$

Source

Drawback: involve operations over the entire state set of the MDP, that is, they require sweeps of the state set.

Asynchronous Dynamic Programming

DP methods described so far used synchronous updates (all states in parallel)

Asynchronous DP:

- backs up states individually, in any order
- can significantly reduce computation
- guaranteed to converge if all states continue to be selected

Asynchronous Dynamic Programming

Three simple ideas for asynchronous dynamic programming:

1. In-place dynamic programming: do not copy V^{π} but make in-place update during iteration method;
2. Prioritised sweeping: select state based on magnitude of Bellman error
3. Real-time dynamic programming: consider only relevant states using samples $< s, a, r, s' >$

These ideas will be extremely useful in a model-free setting!

Background

1. Sutton & Barto, Chapter 1, 3, 4
2. Practical RL course by YSDA, week 1, 2
3. DeepMind course, lectures 1, 3, 4
4. David Silver course, lecture 1, 2, 3

Thank you for your attention!