Winter School 2024
Reinforcement Learning

Elementary Reinforcement Learning

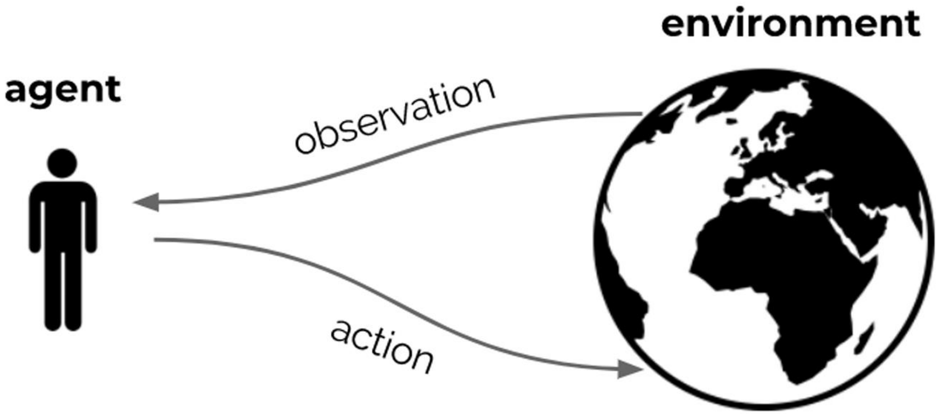
Dr. Lorenzo Brigato

Artificial Intelligence in Health and Nutrition (AIHN) Laboratory
ARTORG Center for Biomedical Engineering Research
University of Bern

Outlook

- Introducing the RL Problem
- Markov Decision Processes (MDPs)
- Exploration and Exploitation

The Agent and the Environment



- At each step t the agent:
  - Receives observation O_t (and reward R_t)
  - Executes action A_t
- The environment:
  - Receives action A_t
  - Emits observation O_{t+1} (and reward R_{t+1})

[Hado van Hasselt, 2021]

Core Concepts

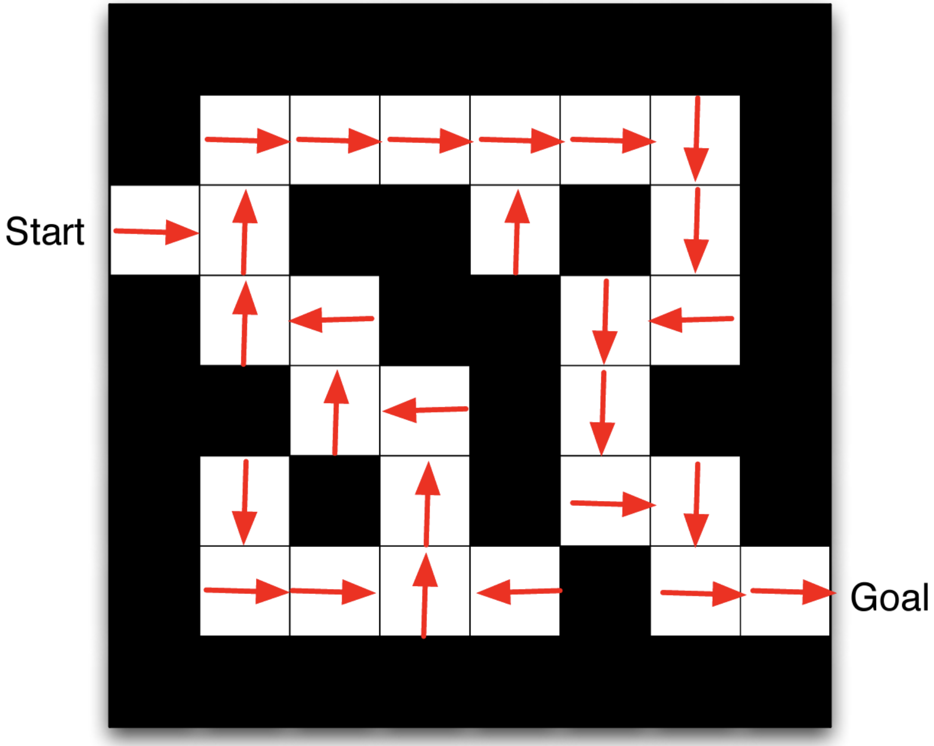
The RL formalism includes:

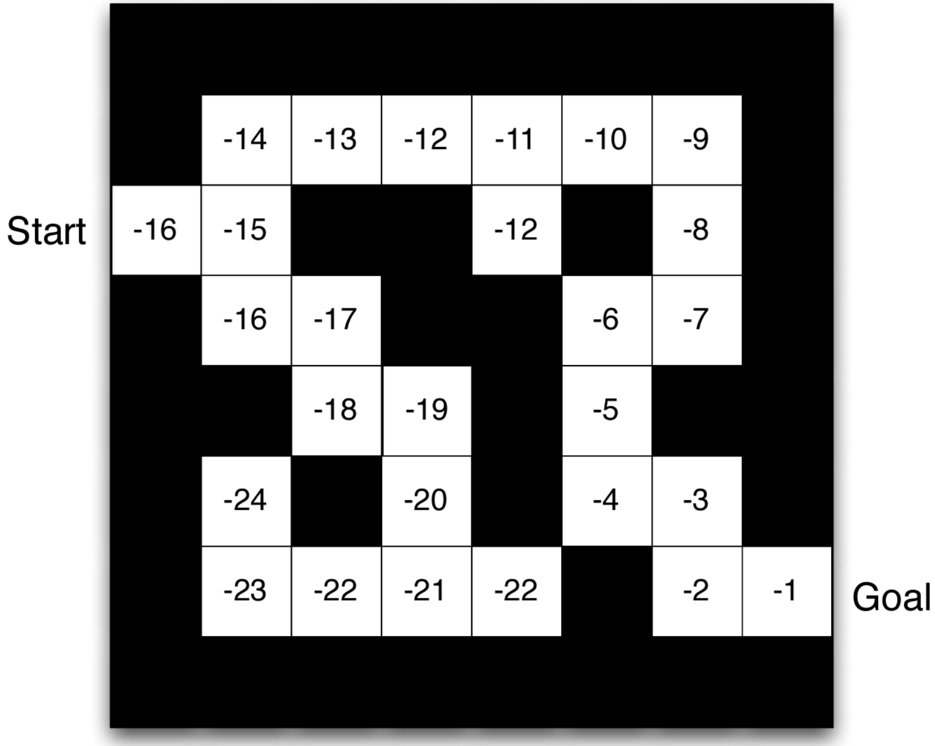
- Reward signal (specifies the goal)
- Environment (dynamics of the problem)
- Agent, containing:
  - Agent state
  - Policy
  - Value function estimate (?)
  - Environment Model (?)

Major Components of an RL Agent

- An RL agent may include one or more of these components:
  - Policy: agent's behavior function
  - Value function: how good is each state and/or action
  - Model: agent's representation of the environment

Maze Example





[Hado van Hasselt, 2021]

Agent Categories (1/2)

- Value Based
  - No Policy (Implicit)
  - Value Function
- Policy Based
  - Policy
  - No Value Function
- Actor Critic
  - Policy
  - Value Function

Agent Categories (2/2)

- Model Free
  - Policy and/or Value Function
  - No Model
- Model Based
  - Optionally Policy and/or Value Function
  - Model

Introduction to MDPs

- MDPs formally describe an environment for RL
- Current state completely characterizes the process (fully observability)
- Almost all RL problems can be formalized as MDPs, e.g.
  - Optimal control primarily deals with continuous MDPs
  - Partially observable problems can be converted into MDPs (POMDPs)
  - Bandits are MDPs with one state

Information or Markov State

– An information state (a.k.a. Markov state) contains all useful information from the history.

Definition

A state S_t is Markov if and only if:

$$\mathbb{P}[S_{t+1}|S_t] = \mathbb{P}[S_{t+1}|S_1 \dots S_t]$$

– "The future is independent of the past given the present"
– Doesn't mean it contains everything, just that adding more history doesn't help
– Once the state is known, the history may be thrown away

Markov Decision Process

– A Markov decision process (MDP) is an environment in which all states are Markov and defined as follows

Definition
A Markov Process (or Markov Chain) is a 4-tuple $< \mathcal{S}, \mathcal{A}, \mathcal{P}, \mathcal{R}, \gamma >$ • $\mathcal{S}$ is a (finite) set of states   • $\mathcal{A}$ is a (finite) set of actions   • $\mathcal{P}$ is a state transition matrix, i.e., $\mathcal{P}_{ss'}^{a} = \mathbb{P}[S_{t+1} = s' \mid S_t = s,\ A_t = a]$ • $\mathcal{R}$ is a reward function, $\mathcal{R}_s^a = \mathbb{E}[R_{t+1} \mid S_t = s,\ A_t = a]$ • γ is a discount factor, $\gamma \in [0, 1]$

State Transition Matrix

- For a Markov state s, an action a, and a successor state s′, the state transition probability is defined by

$$\mathcal{P}_{ss'}^{a} = \mathbb{P}[S_{t+1} = s' \mid S_t = s, At = a]$$

- A State transition matrix $\mathcal{P}$ can also be defined that holds the transition probabilities from all state-action couples (s, a) to all successor states s′

$$\mathcal{P} = \begin{bmatrix} \mathcal{P}_{11} & \cdots & \mathcal{P}_{1n} \\ \vdots & \ddots & \vdots \\ \mathcal{P}_{n1} & \cdots & \mathcal{P}_{nn} \end{bmatrix}$$

each row of the matrix sums to 1.

Return

Definition

The return G_t is the total discounted reward from time-step t:

$$G_t = R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+3} + \cdots = \sum_{k=0}^{\infty} \gamma^k R_{t+k+1}$$

- 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 $\gamma^k R$.
- This values immediate reward above delayed reward.
  - γ close to 0 leads to “myopic” evaluation
  - γ close to 1 leads to “far-sighted” evaluation

Policies

Definition

A policy π is a distribution over actions given states:

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

– A policy fully defines the behaviour of an agent
– MDP policies depend on the current state (not the history)

State-Value and Action-Value Functions

Definition

The state-value function $v_\pi(s)$ of an MDP is the expected return starting from state s, and then following policy π:

$$v_\pi(s) = \mathbb{E}_\pi[G_t \mid S_t = s]$$

Definition

The action-value function $q_\pi(s, a)$ of an MDP is the expected return starting from state s, taking action a, and then following policy π:

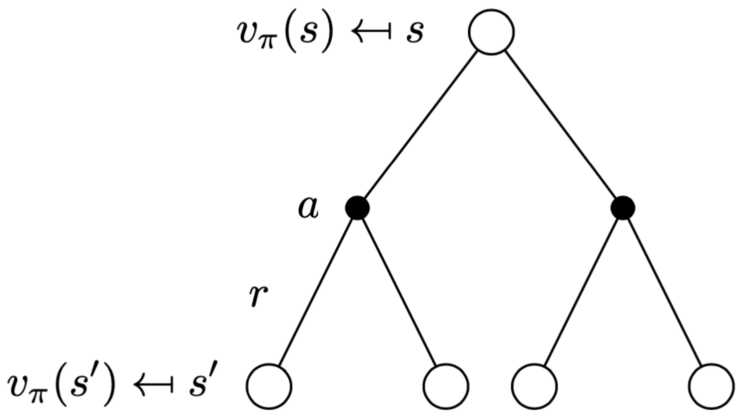
$$q_\pi(s, a) = \mathbb{E}_\pi[G_t \mid S_t = s, \mathrm{A}_t = a]$$

Bellman Expectation Equation

- The value function can be decomposed into two parts:
  - Immediate reward R_{t+1}
  - Discounted value of successor state $\gamma v_\pi(S_{t+1})$ under policy π

$$
\begin{aligned}
v_\pi(s) &= \mathbb{E}_\pi[G_t \mid S_t = s] \\
&= \mathbb{E}_\pi[R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+3} + \cdots \mid S_t = s] \\
&= \mathbb{E}_\pi[R_{t+1} + \gamma (R_{t+2} + \gamma R_{t+3} + \cdots) \mid S_t = s] \\
&= \mathbb{E}_\pi[R_{t+1} + \gamma G_{t+1} \mid S_t = s] \\
&= \mathbb{E}_\pi[R_{t+1} + \gamma v_\pi(S_{t+1}) \mid S_t = s] \\
&= \mathbb{E}_\pi[R_{t+1} \mid S_t = s] + \gamma \mathbb{E}_\pi[v(S_{t+1}) \mid S_t = s]
\end{aligned}
$$

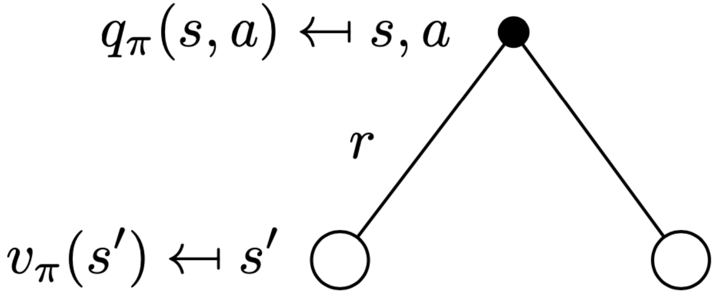
Bellman Expectation Equation for v_π



$$v_\pi(s) = \sum_{a \in \mathcal{A}} \pi(a \mid s) \left(\mathcal{R}_s^a + \gamma \sum_{s' \in \mathcal{S}} \mathcal{P}_{ss'}^a v_\pi(s') \right)$$

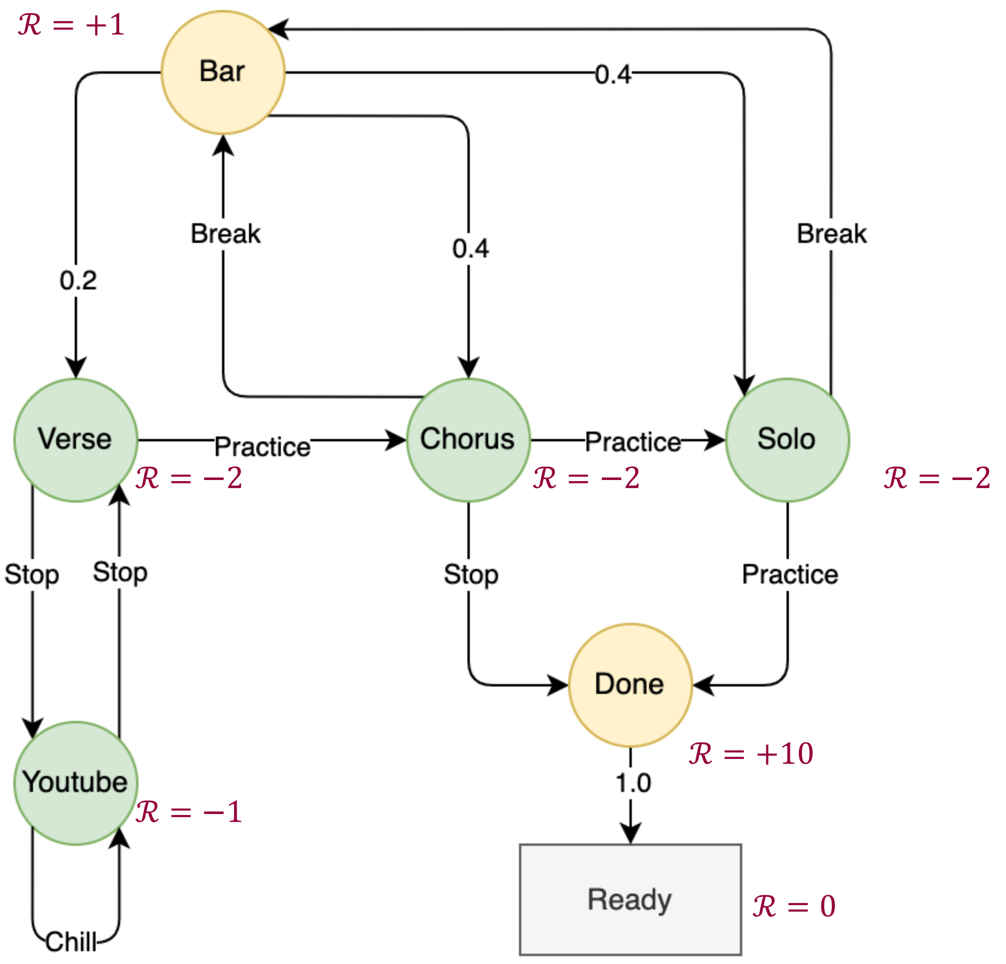
$$v_\pi(s) = \sum_{a \in \mathcal{A}} \pi(a \mid s)\, \mathcal{R}_s^a + \gamma \sum_{a \in \mathcal{A}} \pi(a \mid s) \left(\sum_{s' \in \mathcal{S}} \mathcal{P}_{ss'}^a v(s') \right)$$

Bellman Expectation Equation for q_π



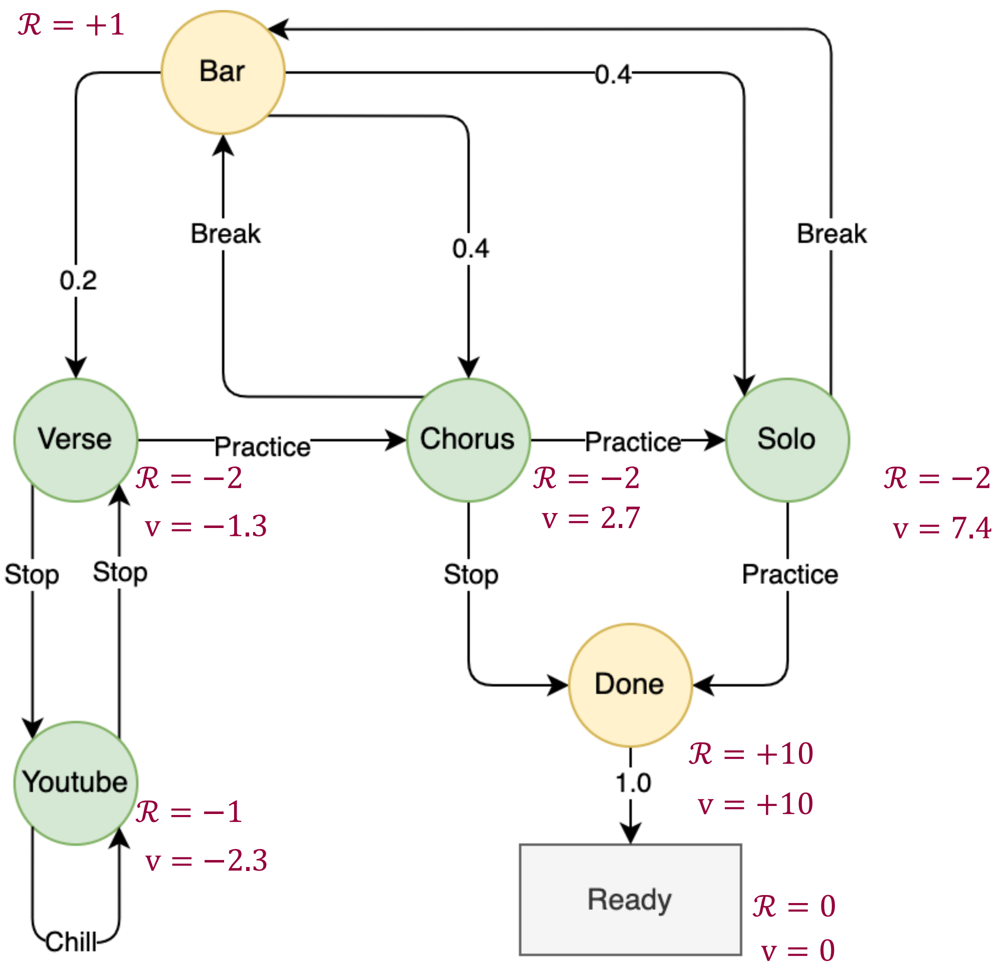
$$q_\pi(s,a) = \mathcal{R}_s^a + \gamma \sum_{s' \in \mathcal{S}} \mathcal{P}_{ss'}^a \mathrm{v}_\pi(s')$$

Example: Song Learning MDP



Example: Bellman Expectation Equation in MDP

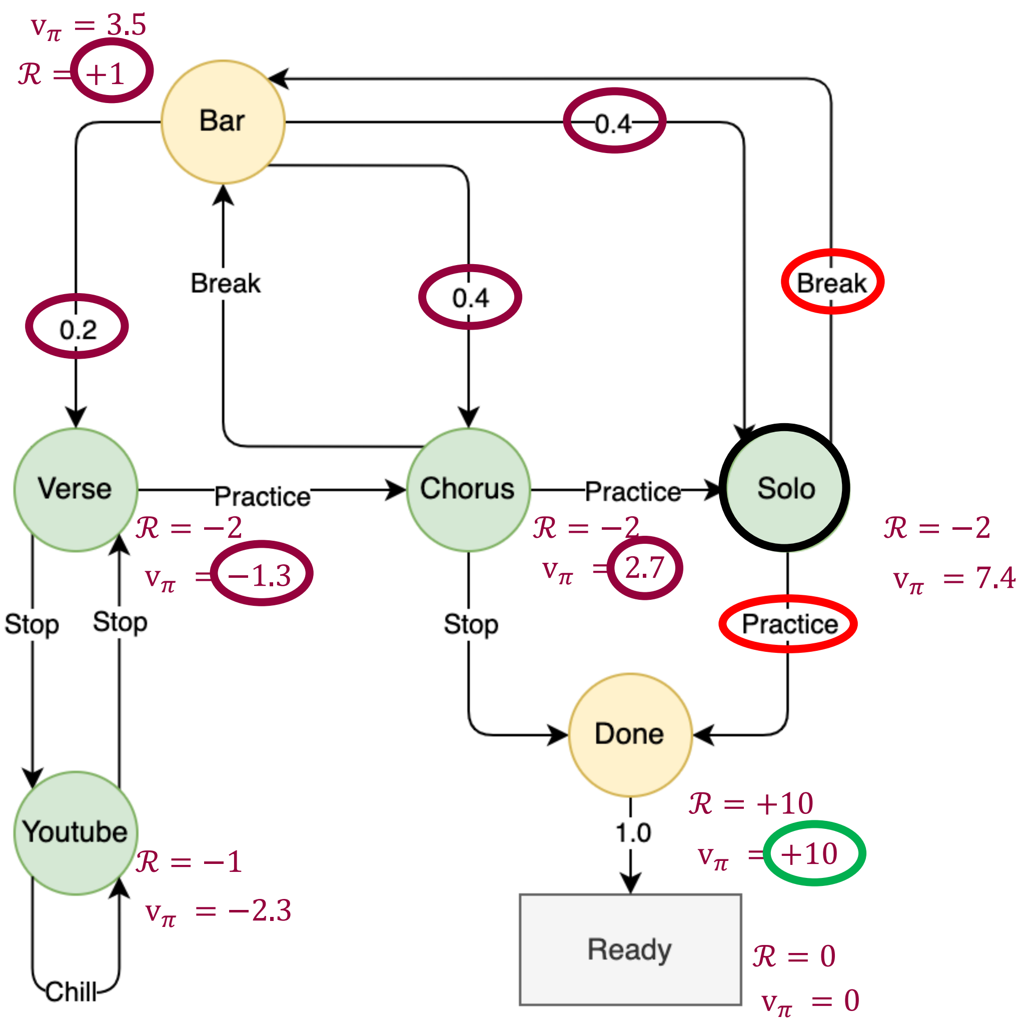
$v_\pi(s)$ for $\pi(a|s) = 0.5$



Example: Bellman Expectation Equation in MDP

$v_\pi(s)$ for $\pi(a|s) = 0.5$

$$7.4 = 0.5 * (1 - 0.2 * 1.3 + 0.4 * 2.7 + 0.4 * 7.4) + 0.5 * 10$$



Bellman Expectation Equation (Matrix Form)

- The Bellman equation is a linear equation

$$\mathrm{v}_{\pi} = \mathcal{R}^{\pi} + \gamma \mathcal{P}^{\pi} \mathrm{v}$$

- It can be solved directly

$$\mathrm{v}_{\pi} = (\mathrm{I} - \gamma \mathcal{P}^{\pi})^{-1} \mathcal{R}^{\pi}$$

- Matrix Inversion is computational heavy $\mathcal{O}(n^3)$
- Direct solution only possible for small MDPs
- There are many iterative methods for large MDPs, e.g.
  - Dynamic programming
  - Monte-Carlo evaluation
  - Temporal-Difference learning

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

Exploration vs. Exploitation Dilemma

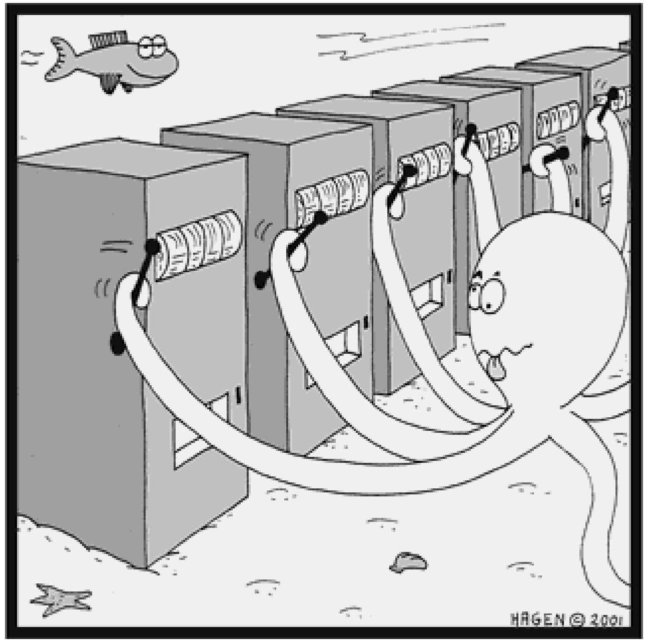
- Online decision-making involves a fundamental choice:
  - Exploitation Make the best decision given current information
  - Exploration Gather more information
- The best long-term strategy may involve short-term sacrifices
- Gather enough information to make the best overall decisions

Principles

- Naive Exploration
  - Add noise to greedy policy (e.g., ε-greedy)
- Optimistic Initialization
  - Assume the best until proven otherwise
- Optimism in the Face of Uncertainty
  - Prefer actions with uncertain values

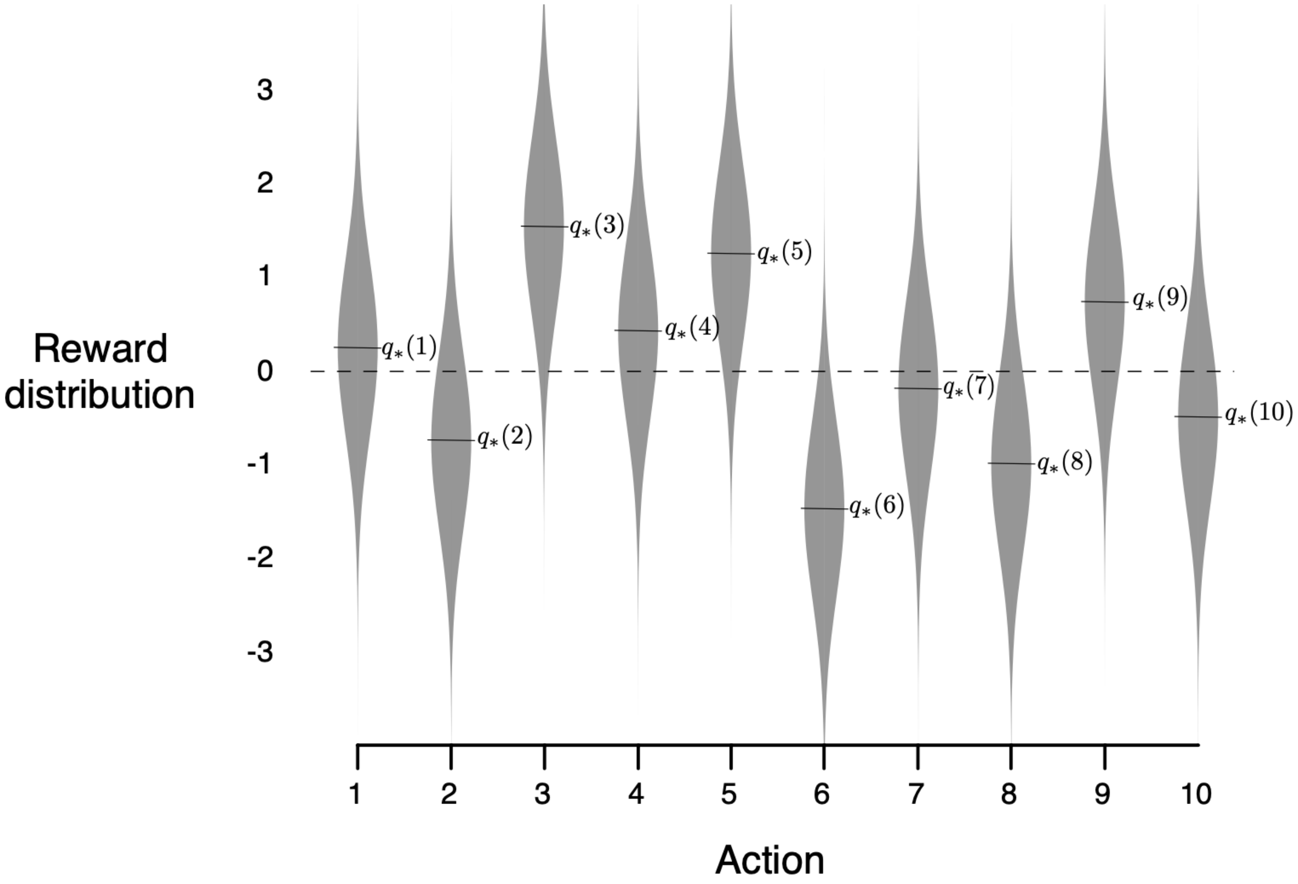
Multi-Armed Bandits

- A multi-armed bandit is a set of distributions $< \mathcal{A}, \mathcal{R} >$
- $\mathcal{A}$ is a (known) set of actions (or “arms")
- $\mathcal{R}^a(r) = \mathbb{P}[r|a]$ is an unknown probability distribution over rewards
- At each step t the agent selects an action $a_t \in \mathcal{A}$
- The environment generates a reward $r_t \sim \mathcal{R}^{a_t}$
- The goal is to maximize cumulative reward $\sum_{\tau=1}^{t} r_\tau$



[David Silver, IRL, UCL 2015]

10-armed Testbed



[An Introduction to Reinforcement Learning, Sutton and Barto]

Greedy Algorithm

- One of the simplest algorithm
- Select action with highest value:

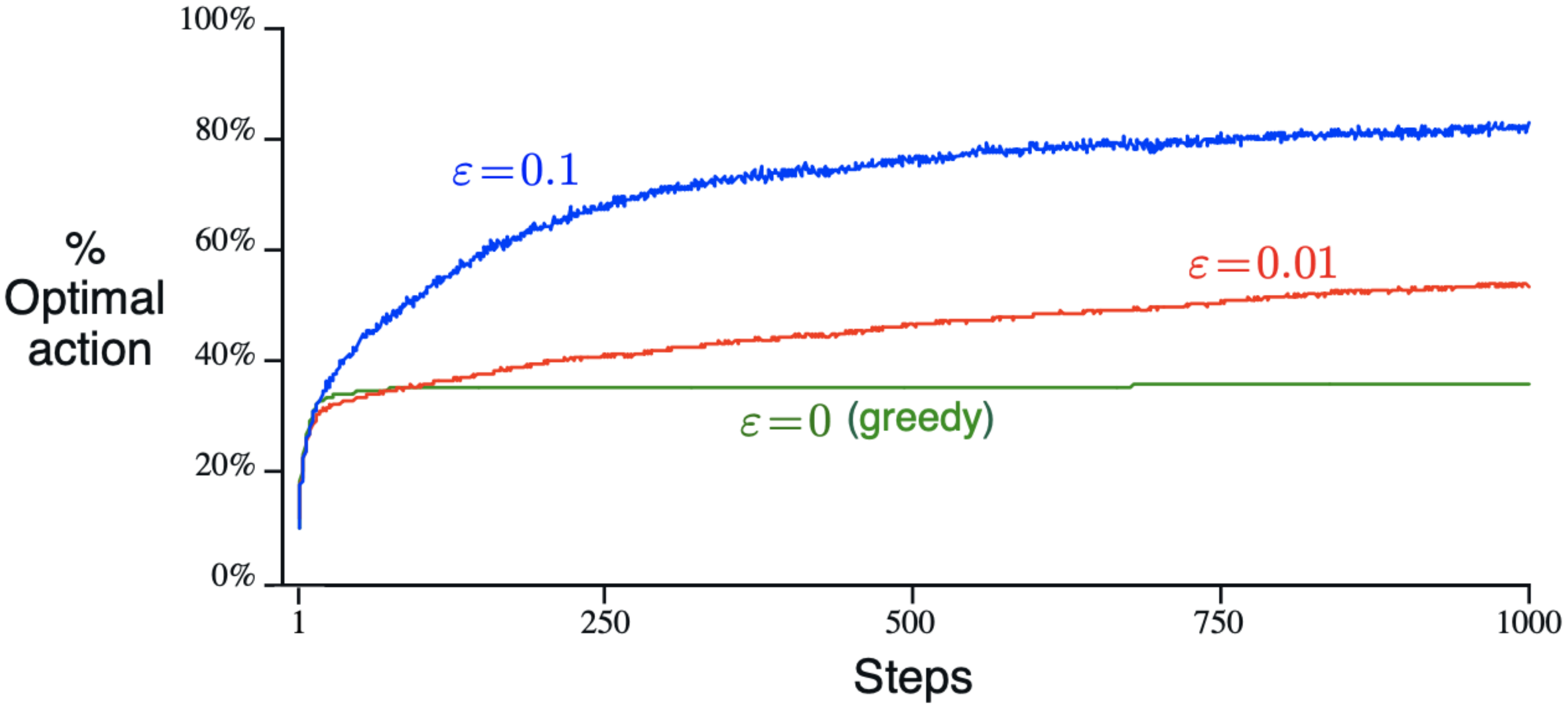
$$A_t = \underset{a \in \mathcal{A}}{argmax}\, Q_t(a)$$

- Greedy can lock onto a suboptimal action forever

ε-Greedy Algorithm

- The ε-greedy algorithm:
  - With probability 1 $-$ε select greedy action: $A_t = \underset{a \in \mathcal{A}}{argmax}\, Q_t(a)$
  - With probability ε select a random action
- ε-greedy continues to explore

Greedy vs ε-Greedy



[An Introduction to Reinforcement Learning, Sutton and Barto]

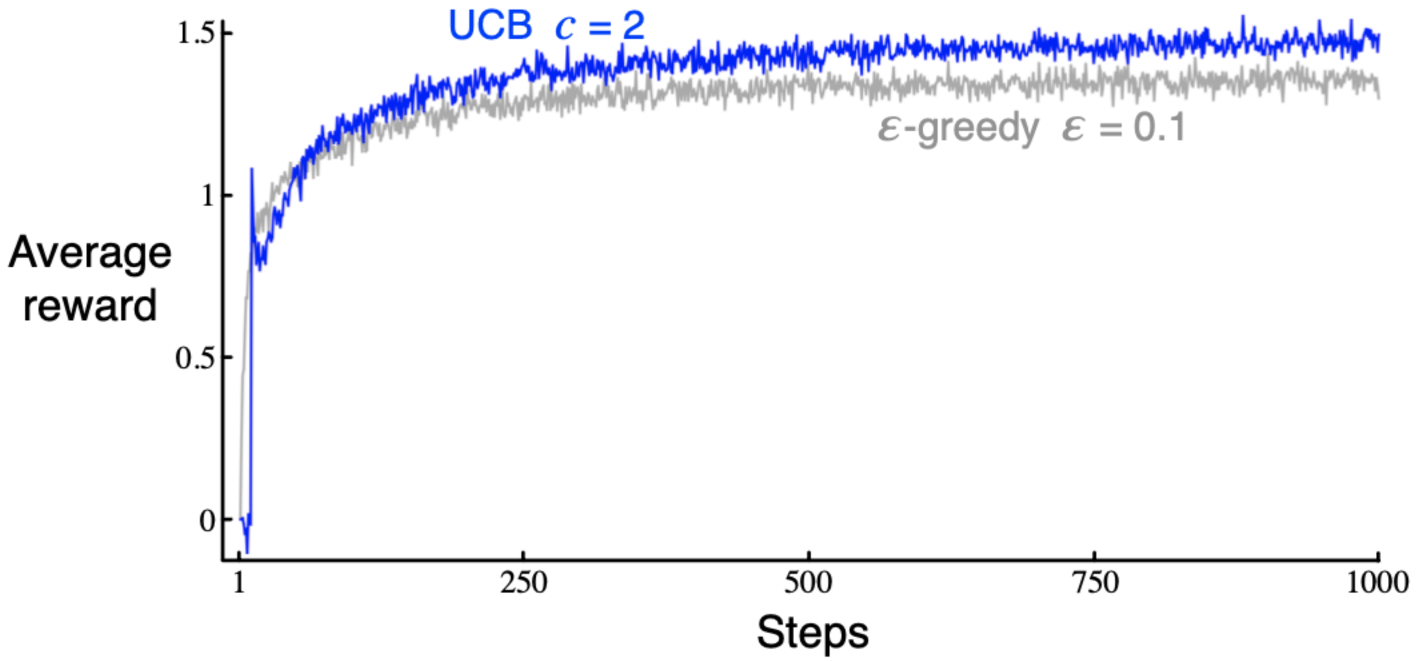
Upper Confidence Bound (UCB)

- The ε-Greedy algorithm performs exploration without any preference (random).
- Why not explore in a more explicit way?
- UCB selects the actions with the most uncertain value function estimates!

$$A_t \doteq \underset{a}{\operatorname{argmax}} \left[Q_t(a) + c \sqrt{\frac{\ln t}{N_t(a)}} \right]$$

- $N_t(a)$ denotes the times action a was selected prior to time t
- Eventually, the square-root term is a measure of uncertainty

UCB vs ε-Greedy



[An Introduction to Reinforcement Learning, Sutton and Barto]

Issues in Exploration

- Differently that in Bandits we have:
  - States (usually very large)
  - Sometimes sparse / long-term reward
  - Function approximation
- We can apply lessons from simple Bandits but also conceive more general strategies

Intrinsic Reward

- Augment the reward with an additional (vanishing) reward term

$$R^t{}_t = R^e{}_t + \beta\, R^i{}_t$$

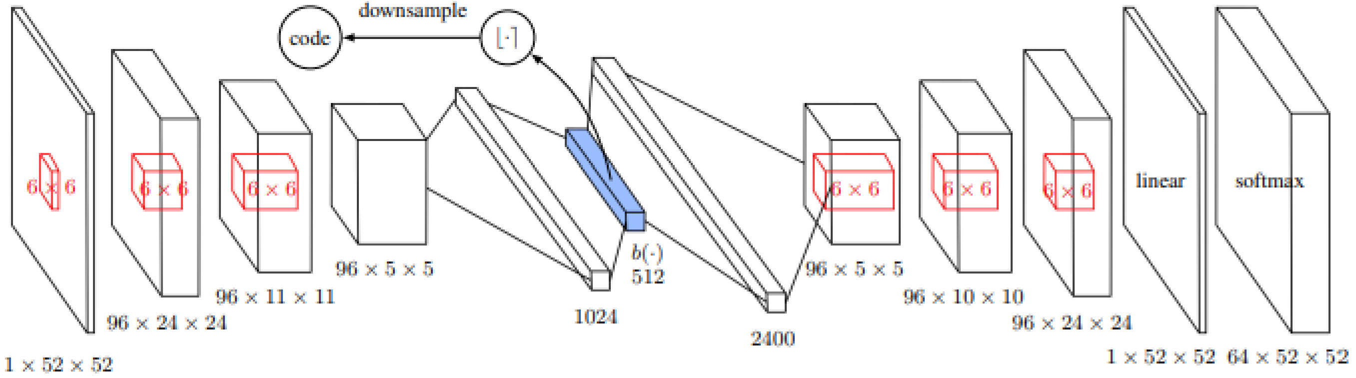
- with $R^e{}_t$ being the extrinsic reward (task reward) and $R^i{}_t$ the intrinsic reward (exploration bonus)
- You can run any algorithm using the new reward $R^t{}_t$

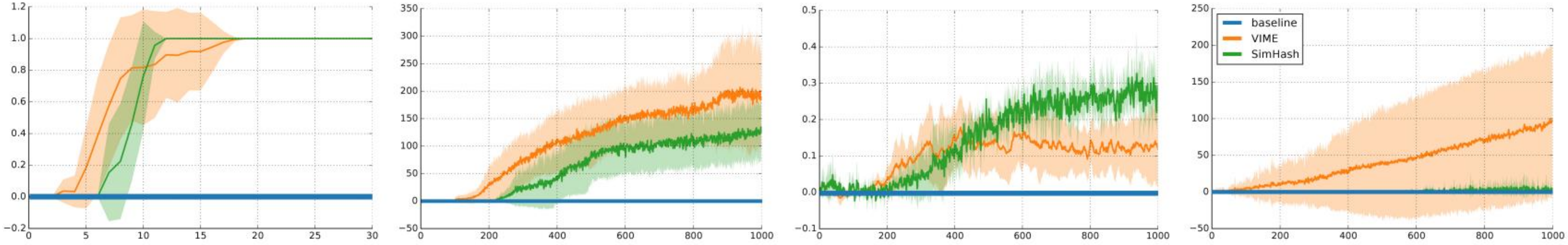
Intrinsic Reward

- How can we define the intrinsic reward bonus? Several options (on-going research):
  - Discover new states
  - Improve knowledge
  - Improve controllability
  - …

- Some of the approaches:
  - Count-based bonus
  - Prediction-based bonus
  - Empowerment bonus

Count-based





(a) MountainCar (b) CartPoleSwingup (c) SwimmerGather (d) HalfCheetah

[#Exploration: A Study of Count-Based Exploration for Deep Reinforcement Learning, Tang et al.]

Prediction-based

- Computational Curiosity idea: Let’s explore to improve skills
- Look for novelty and surprises
- Execute behaviors that reduce uncertainty on how the world works looking for novelty and surprises

- It implies a world-model (model-based RL)
  - predict what is going to happen given what I am expecting to happen

Prediction-based

- Example: Add as bonus the error in the prediction. Bigger error in prediction means less knowledge of the next state (= increased novelty)
  - Given an encoding of the state $\phi(s)$ the agent learns a prediction model:

$$f: (\phi(s), at) \text{ -> } \phi(s_{t+1})$$

  - Use prediction error e_t (properly normalized and scaled) as exploration bonus $R^i{}_t$:

$$R^i{}_t \propto et = ||\phi(s_{t+1}) - f(\phi(s), at)||^2$$

Prediction-based

- Problem with previous example: Predicting every possible change in the transitions is difficult and may not be necessary
  - E.g., predictions that do not depend on agent actions

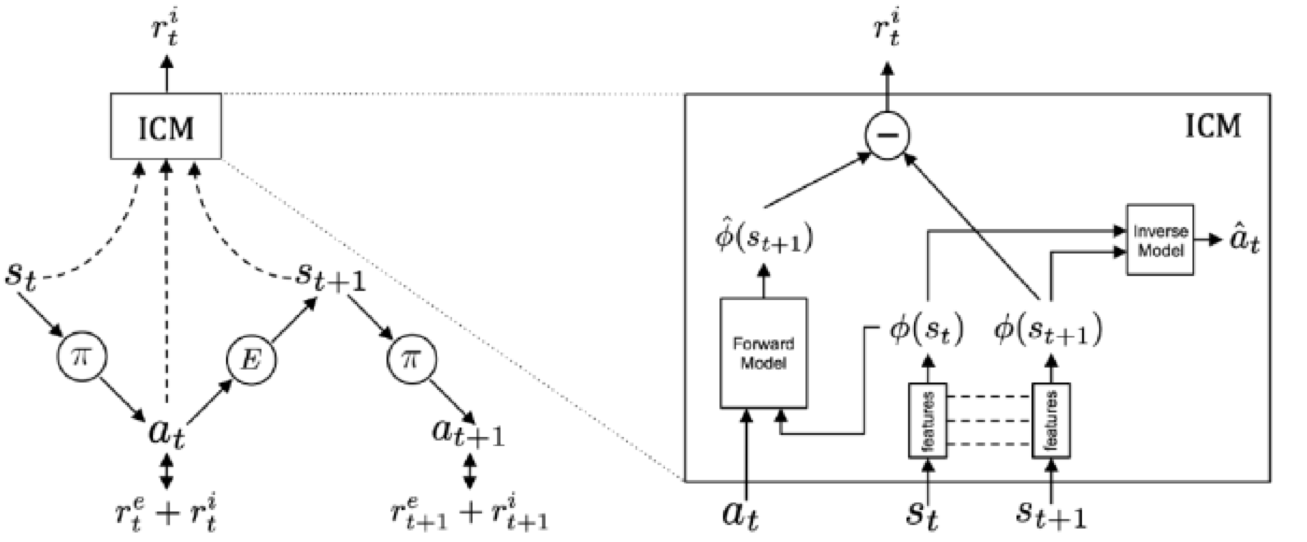
- Proof-of-fact examples:
  - Agent can’t predict TV schedule, so it gets stuck behind the TV
  - Agent can’t predict the random movements of leafes due to wind, so it gets stuck looking at trees
  - …

Prediction-based

- Predict changes that depend on agent's actions, ignore the rest
  - The features of the state depend on the inverse model

Inverse dynamics: $h : (\phi(s_t), \phi(s_{t+1})) \mapsto \widehat{a}_t$
Forward dynamics: $f : (\phi(s_t), a_t) \mapsto \widehat{\phi}(s_{t+1})$

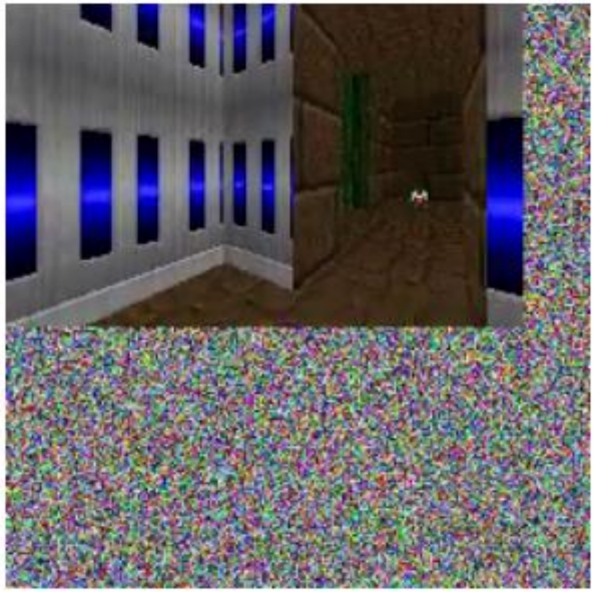
Intrinsic reward:
$r_t^i = \|\widehat{\phi}(s_{t+1}) - \phi(s_{t+1})\|_2^2$



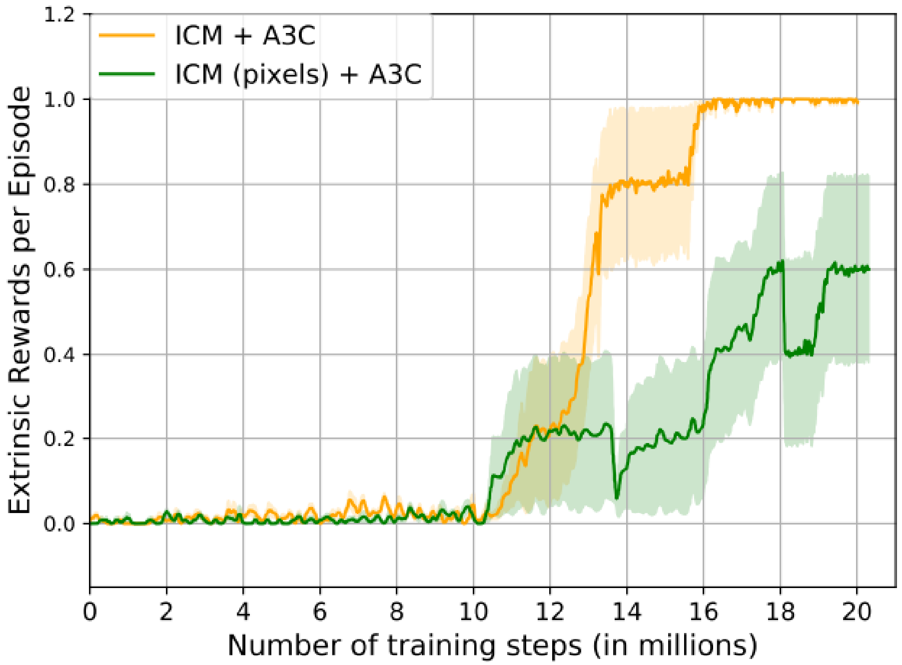
[Curiosity-driven Exploration by Self-supervised Prediction, Pathak et al.]

Prediction-based

- Test: As TV is not controllable by the agent, the model will be blind to the features of the TV



(b) Input w/ noise



[Curiosity-driven Exploration by Self-supervised Prediction, Pathak et al.]