535514: Reinforcement Learning

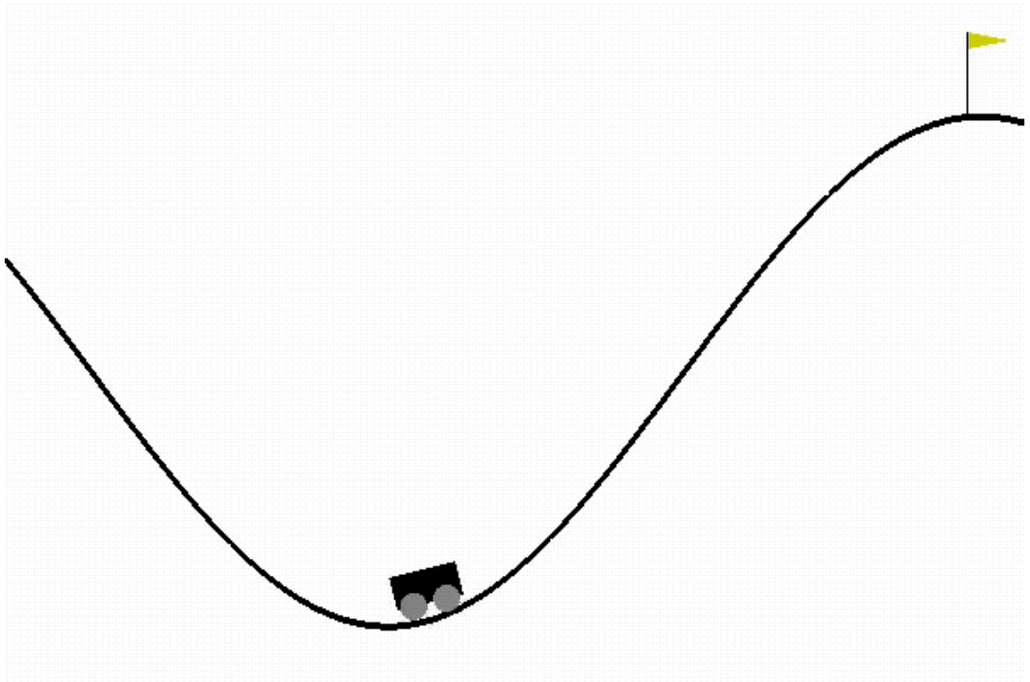
Lecture 1 — Introduction to RL

Ping-Chun Hsieh

February 19, 2024

What is RL? A Motivating Example

- Mountain Car: Navigate the car towards the goal
  - 3 actions available: Drive right / Drive left / Do nothing



- Question: How to choose actions?
- Cast as a *linear control* problem:

  x_t: position at time t

  v_t: velocity at time t

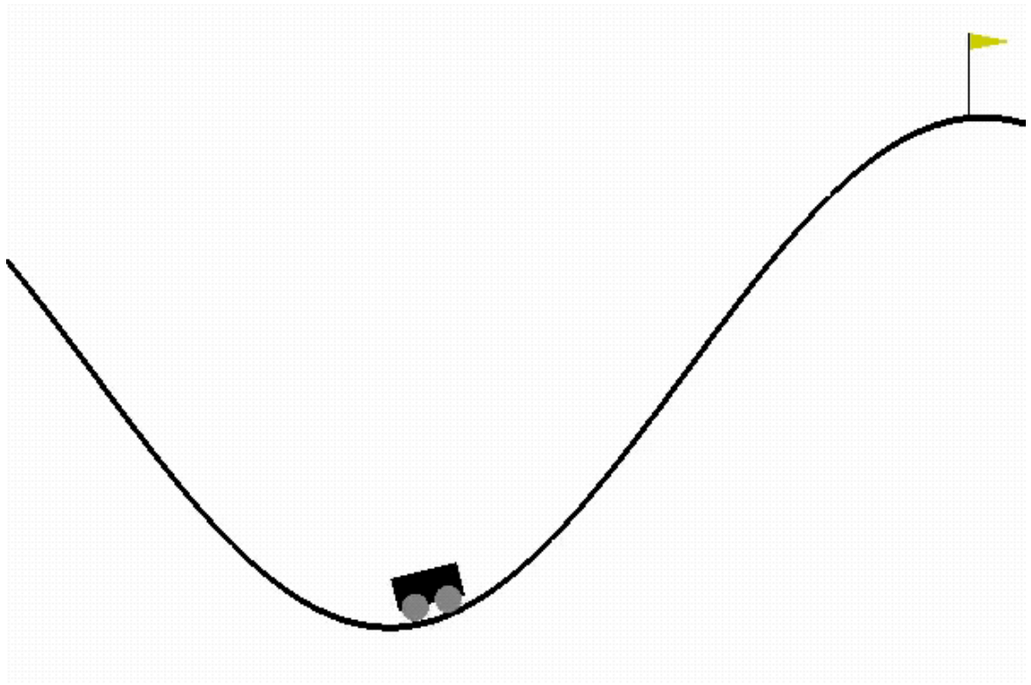
  u_t: force applied to the car at time t

  m: mass of the car

By Newton's motion law: $u_t = m\frac{dv_t}{dt}$

- Given initial condition x_0, v_0, we can make a (deterministic) plan towards the goal
- This is usually called *optimal control*

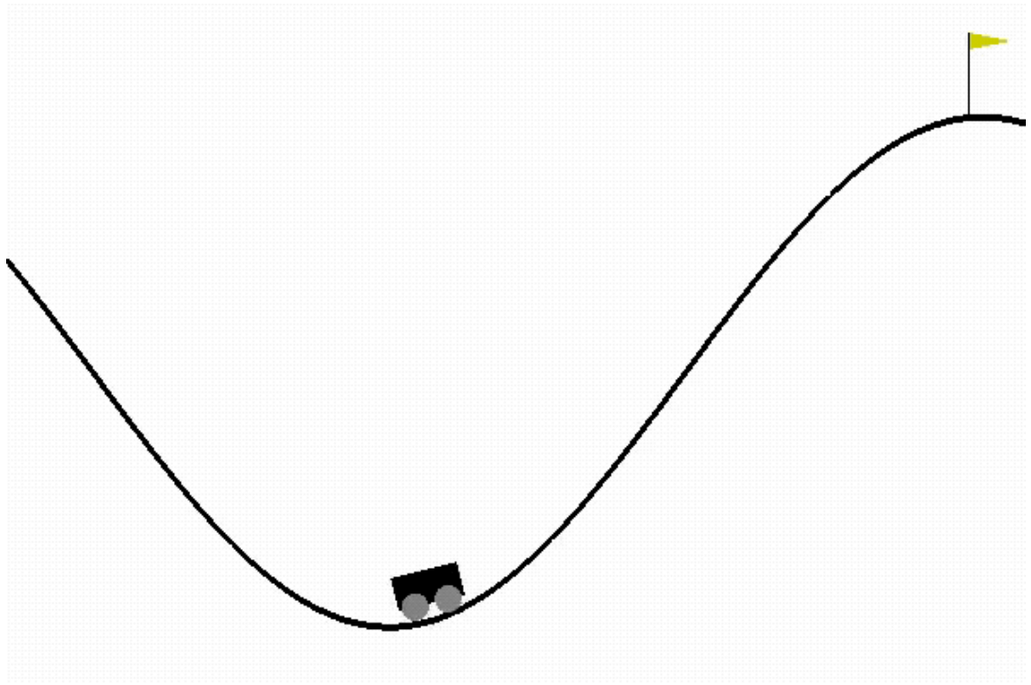
What is RL? A Motivating Example (Cont.)



- Question: What if our motion model is not (fully) known a priori?
- Example: unknown friction

- We can still use Newton's motion law, but need to learn the amount of friction from observations (= reconstruction of a Markov process)
- This is usually called *adaptive control* in control community
- In RL community, this is a *model-based RL* problem

What is RL? A Motivating Example (Cont.)



- Question: What if our motion model is not (fully) known a priori and it is too complex to reconstruct?

- Instead of building model, we just keep playing with the car and learn from the interactions with the environment
- In RL community, this is a *model-free RL* problem

RL ≈ Learning-based control

(Learn to take a sequence of actions to achieve a specific goal in an unknown environment)

- "Model-based" RL: decompose RL into 2 subproblems
  1. Learning a model of the environment
  2. Planning based on this model

- "Model-free" RL:
  - Learning a policy *without* constructing a model
  - Instead, model-free RL uses a proxy called "*value function*"

How to Specify a Goal?

- A natural idea: consequences of actions are additive

- Examples:

Atari Games: *Breakout*

Maze navigation

How to Specify a Goal?

- **Reward hypothesis**: every goal can be specified as an optimization problem in terms of expected total rewards
  - Reward = a “scalar” feedback signal
  - An indicator of how good an instantaneous action is

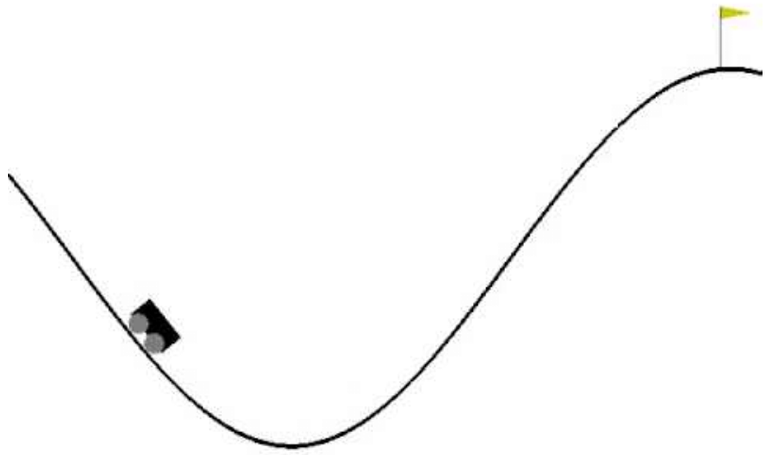
- Remark: A historical reason for this: optimal control uses “*cost*”, which is the negative version of reward

- Question: Do you like this hypothesis? Any limitations?

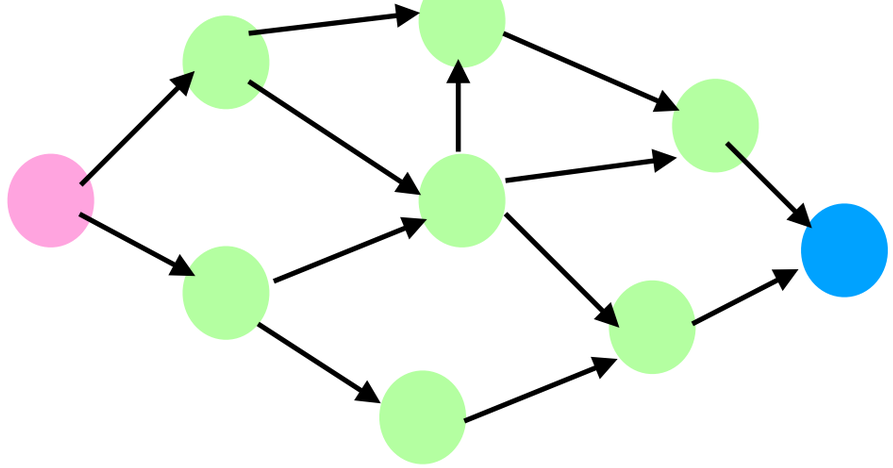
Online discussion: http://incompleteideas.net/rlai.cs.ualberta.ca/RLAI/rewardhypothesis.html

Examples: Goals Specified By Rewards

- Mountain car: reach the target position
  - reward = $+1$ at the target position; 0 elsewhere



- Packet routing: minimize total delay
  - reward = — (transmission time at each hop)



- Recommender system: maximize click-through rate
  - reward = $+1$ if user clicks; 0 otherwise

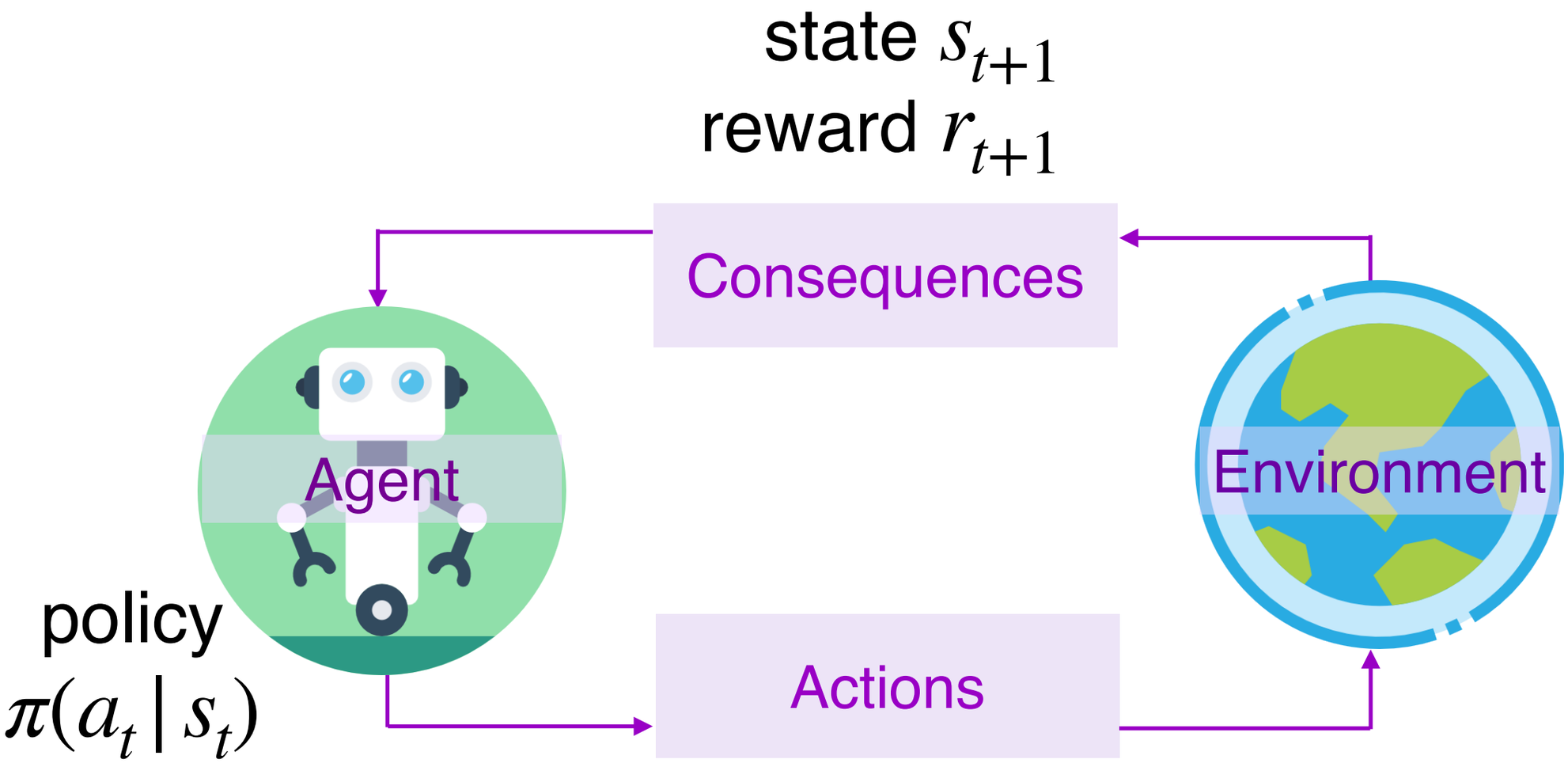
What if Reward Signals are not Available?

- Imitation learning (discussed in Week 14)
  - Learning from expert demonstrations in a supervised manner
  - Infer the reward function by expert demonstrations (aka inverse RL)

- Reward-free RL
  - “Unsupervised” RL in the training phase only from state-action pairs (without any reward signal)
  - Reward signal specified later in the testing phase

- RL from human feedback (RLHF) (discussed in Week 15)
  - Learn a “reward function” from human preferences

RL Interaction Protocol



a_t

Trajectory: $s_0, a_0, r_1, s_1, a_1, r_2, \cdots$

Yann LeCun's Cake Analogy

- Suppose machine intelligence is like a cake:

Yann LeCun

Reinforcement learning (cherry)

Supervised learning (icing)

Self-supervised learning (the bulk of cake)

- How shall RL researchers reply to this?

Another Cake In the Eyes of RL Researchers

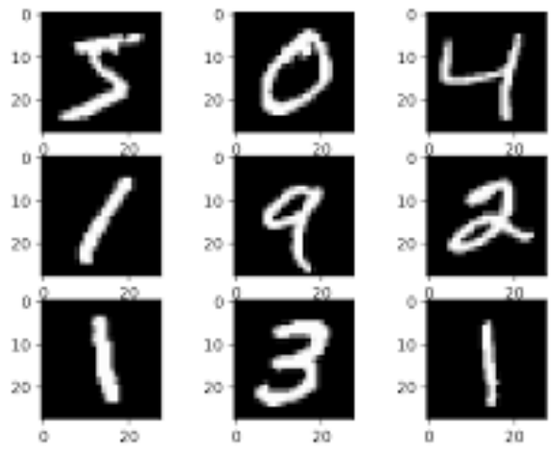
- There could be many cherries on the same cake lol

- Question: But what does the “cake analogy” actually mean?

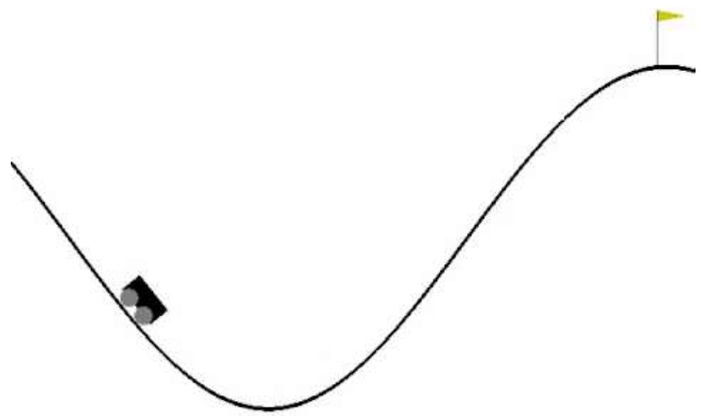
https://medium.com/syncedreview/yann-lecun-cake-analogy-2-0-a361da560dae

Features of RL (Compared to SL)

SL



RL

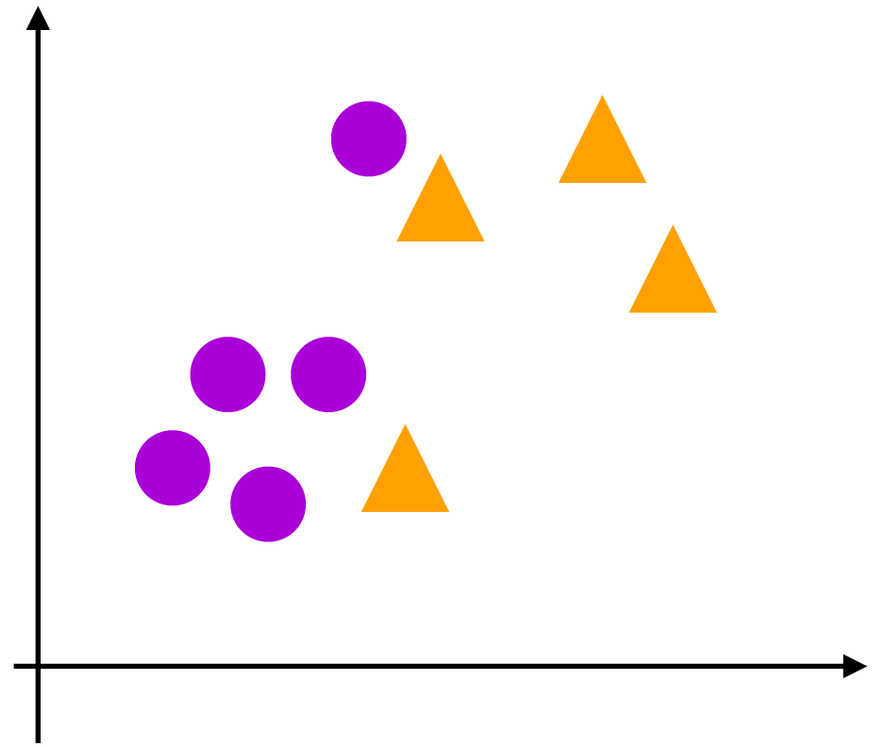


1. No labels, only reward signal
2. Non-i.i.d. data (if we view (s_t, a_t, r_{t+1}) as one data sample)
3. Actions taken will affect the data collected in the future
4. Exploration available and more importantly, required!
5. *Thinking* and *doing* need not be aligned (called off-policy)

Recall: Supervised Learning (SL)

- Data: feature-label pairs
  $\{(s_1, a_1^*), (s_2, a_2^*), \cdots\}$
- Learn a classification policy $\pi_\theta(a \mid s)$
  - e.g.: Sigmoid for binary classification

$$\pi_\theta(a \mid s) = \frac{1}{1 + \exp(-\theta^\intercal s)}$$



- Maximize log-likelihood of the data (=maximum likelihood estimation)

$$\theta^* = \arg\max_\theta \sum_n \log \pi_\theta(a_n^* \mid s_n)$$

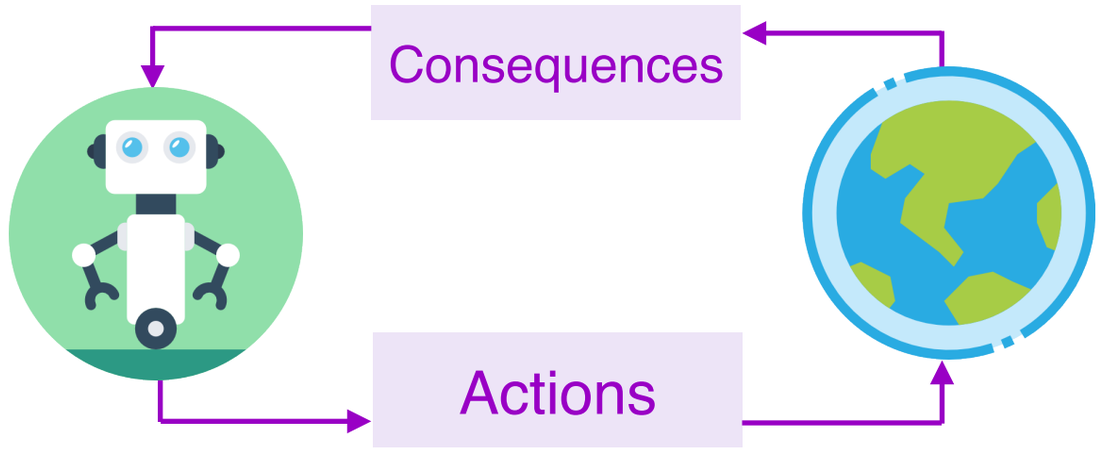
- Gradient update:

$$\theta_{n+1} \leftarrow \theta_n + \alpha \nabla_\theta \log \pi_\theta(a_n^* \mid s_n)$$

(Slide adapted from Pascal Poupart)

RL as an Analogy of Supervised Learning

- Data: state-action-reward triples
  $\{(s_0, a_0, r_1), (s_1, a_1, r_2), \cdots\}$
- Learn a (stochastic) policy $\pi_\theta(a \mid s)$



- Maximize (discounted) sum of rewards

$$\theta^* = \arg\max_\theta \sum_n \mathbb{E}[r_{n+1} \mid s_n, a_n]$$

- Gradient update (policy gradient):

$$\theta_{n+1} \leftarrow \theta_n + \alpha G_n \nabla_\theta \log \pi_\theta(a_n \mid s_n)$$

(G_n:sum of reward from step n onwards)

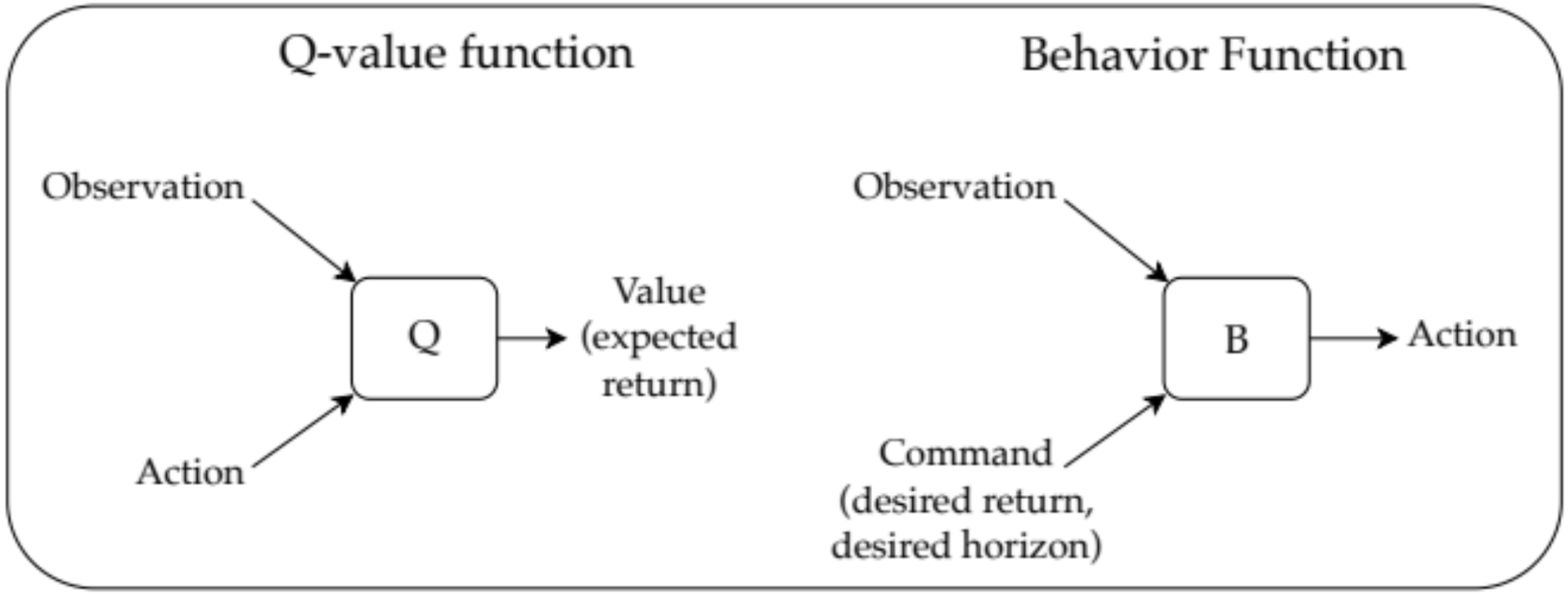
(Slide adapted from Pascal Poupart)

RL via Supervised Learning?

- Subsequently, we will mention an interesting idea on “RL via SL”:

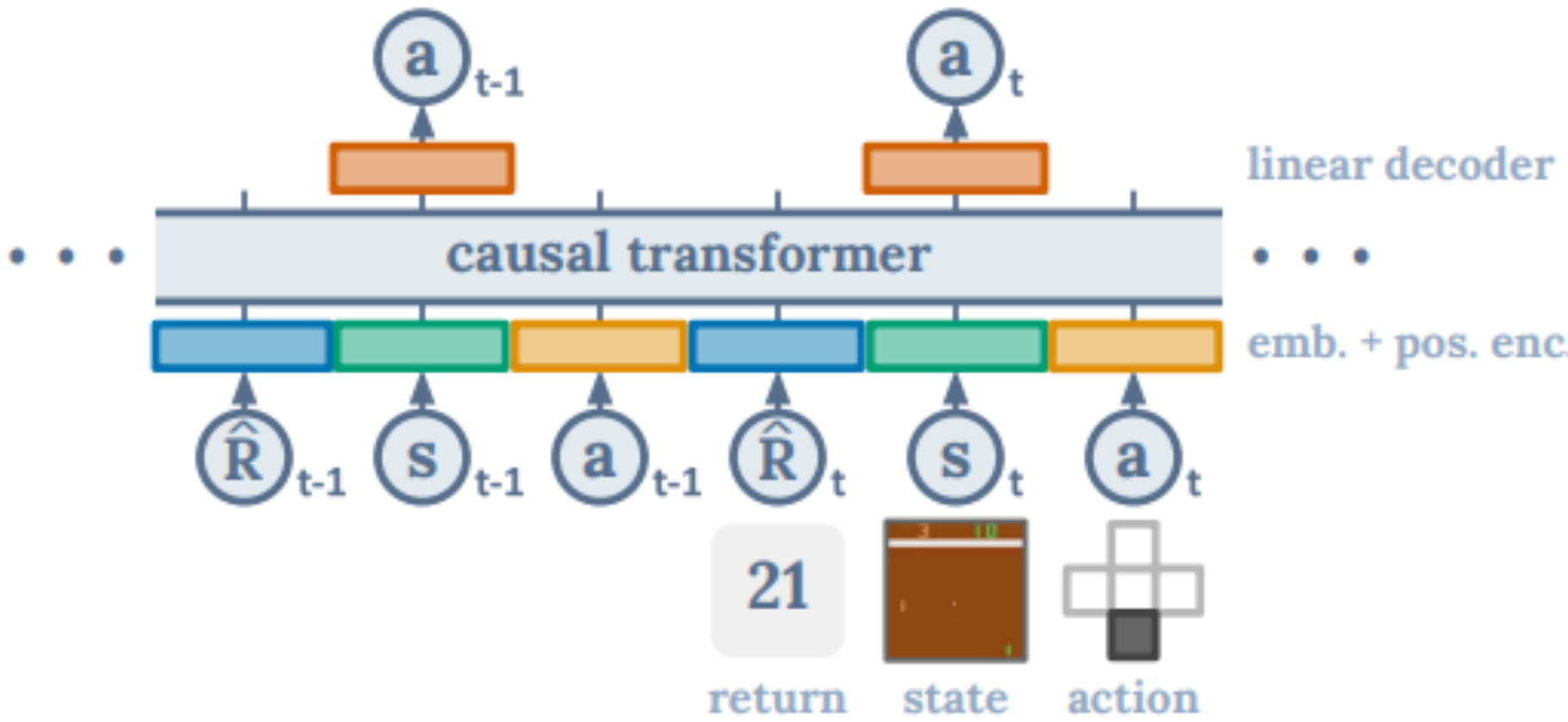
Upside-Down RL

- e.g., https://arxiv.org/pdf/1912.02877.pdf by Jürgen Schmidhuber



A Popular Recent RL Topic: Decision Transformer

- Decision Transformer: An implementation of Upside-Down RL

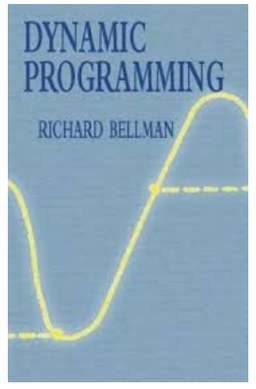


(This framework is particularly useful for Offline RL, i.e., RL from offline data)

Chen et al., Decision Transformer: Reinforcement Learning via Sequence Modeling, NeurIPS 2021

Some Historical Accounts

- 1960s: Dynamic programming (DP) for optimal control
  - Bellman equations and Bellman operators
  - Value functions for MDPs



- 1980s-1990s: Value-based RL in unknown environments
  - Q-learning (Watkins, 1982)
  - TD learning (Sutton, 1988)

- In the meantime, RL is addressed via policy optimization
  - REINFORCE (William, 1992)
  - Actor-critic (Sutton, 2000; Konda and Tsitsiklis, 2000)

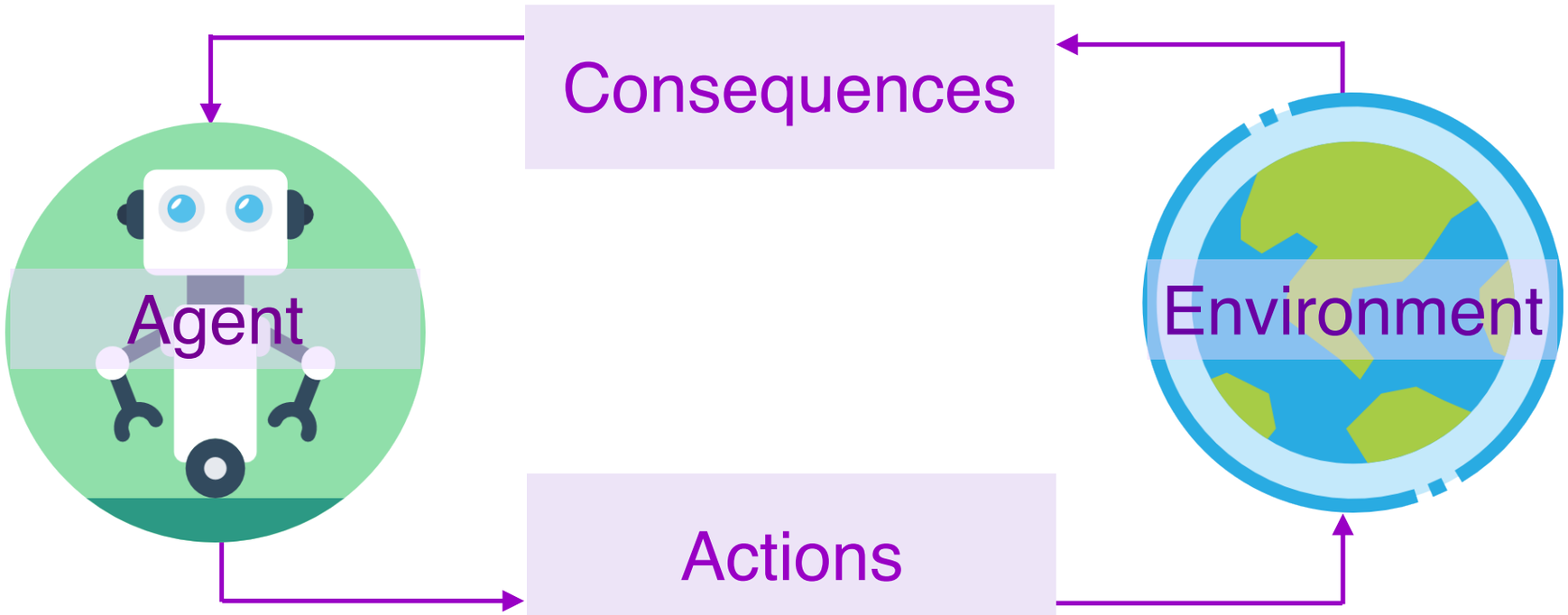
Let’s talk about Markov Decision Processes (MDPs)

Classes of MDPs that We Will Discuss

- Standard MDPs (Lectures 1-3)
  - Foundations of almost all popular RL methods

- Regularized MDPs (Lecture 4)
  - Foundations of Soft Actor Critic (SAC) and Inverse RL

How to Model the Environment?

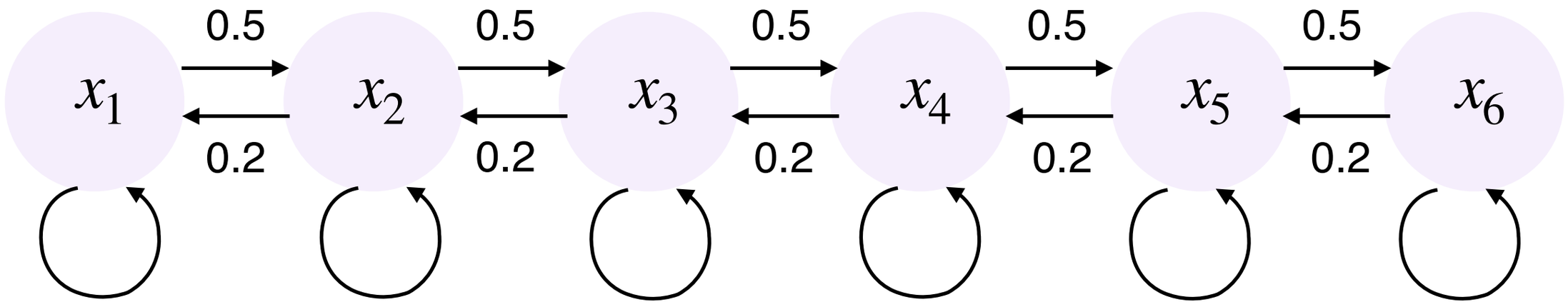


- Various ways to model the environment:
  1. Markov decision processes (MDP) ⟵ Our focus!
  2. Stochastic bandits (aka MDP with only 1 state)
  3. Partially observable Markov decision processes (POMDP)
  4. Adversarial feedback (often used in robust RL & online convex optimization)
  5. …
- Question: Why focusing on MDPs?

Review: Markov Chain

- Example: N-Chain / Mars Rover Problem

[ICML 2000, Strens]



- A sample trajectory is denoted by $s_0, s_1, s_2 \cdots s_t, s_{t+1}, \cdots$
  - e.g. $x_3, x_4, x_5, x_5, x_4, x_3, x_2 \cdots$

- The value on each arrow specifies the transition probability
  - e.g. $\mathbb{P}(s_{t+1} = x_2 \,|\, s_t = x_1) = 0.5$

- Question: Why is it called “Markov”?

Markov Property

- **Markov property:** A state s_t is *Markov* if and only if

$$\mathbb{P}[s_{t+1} \,|\, s_t] = \mathbb{P}[s_{t+1} \,|\, s_1, \cdots, s_t]$$

- Idea: s_t fully captures the statistical information about the full past history $\{s_1, \cdots, s_t\}$

Markov Chain (Formally)

- **Markov Chain**: A Markov chain $(\mathcal{S}, P)$ is specified by:

  1. State space $\mathcal{S}$ = a (finite) set of possible states
  2. Transition matrix $P = [P_{ss'}]$ with elements

$$P_{ss'} = \mathbb{P}[s_{t+1} = s' \,|\, s_t = s]$$

- Question: What are some characteristics of P?

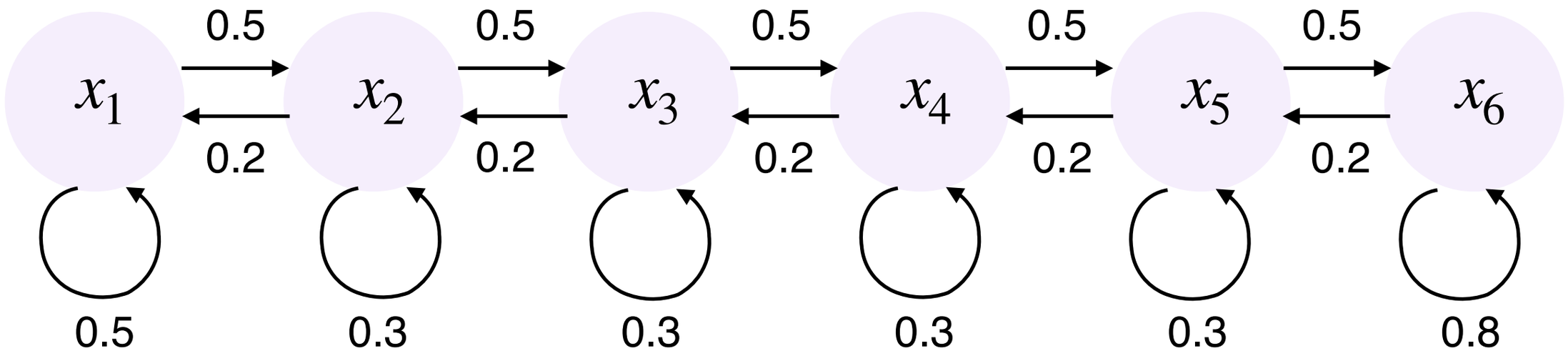
How to indicate whether a state is good or not?

Use *rewards*!

Example: N-Chain / Mars-Rover with Rewards

- Example: N-Chain

[ICML 2000, Strens]



- Reward = 0.05 at x_1, reward = 1 at x_6, and zero otherwise

- A sample trajectory is denoted by $s_0, r_1, s_1, r_2, s_2, r_3 \cdots$
  - e.g. $x_2, 0, x_1, 0.05, x_2, 0$, ...

Markov Reward Process (Formally)

- **Markov Reward Process (MRP)**: An MRP $(\mathcal{S}, P, R, \gamma)$ is specified by

Underlying Dynamics

1. State space $\mathcal{S}$ (assumed finite)
2. Transition matrix $P = [P_{ss'}]$ with $P_{ss'} = \mathbb{P}[s_{t+1} = s' \,|\, s_t = s]$

Task / Goal

3. Reward function $R_s = \mathbb{E}[r_{t+1} \,|\, S_t = s]$
4. Discount factor $\gamma \in [0,1]$

- In this lecture, we shall assume the model parameters P and R_s are known (i.e., no learning)

Return and State-Value Function of an MRP

- Return G_t: Cumulative discounted rewards over a single trajectory from t onwards (random)

$$G_t = r_t + \gamma r_{t+1} + \gamma^2 r_{t+2} + \cdots = \sum_{k=0}^{\infty} \gamma^k r(s_{t+k})$$

- State-value function $V(s)$: Expected return if we start from state s

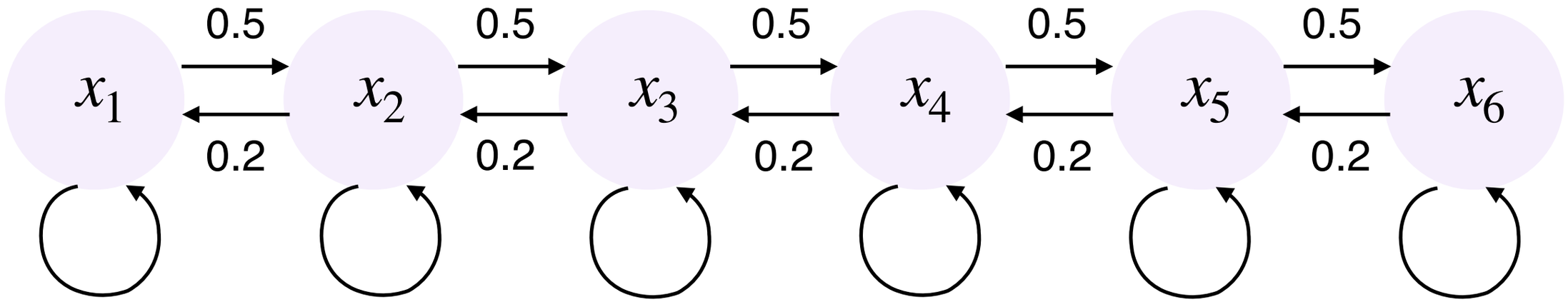
$$V(s) = \mathbb{E}[G_t \,|\, S_t = s]$$

- Remark: $V(s)$ measures the long-term benefit of being in a state

Example: N-Chain / Mars-Rover MRP

- Example: N-Chain

[ICML 2000, Strens]



- Reward = 0.05 at x_1, reward = 1 at x_6, and 0 otherwise

- Sample return for a 5-step episode x_5, x_6, x_6, x_5, x_4 with $\gamma = 0.9$
  - $G_t = 0 + (1 \times 0.9) + (1 \times 0.9^2) + (0 \times 0.9^3) + (0 \times 0.9^4)$

How to Compute $V(s)$ for MRPs?

1. Brute force: Monte-Carlo simulation
   - Draw K trajectories for each starting state s
   - Empirical average return $\approx V(s)$, for large K
2. Recursion: Use dynamic programming

$$
\begin{aligned}
V(s) &= \mathbb{E}[G_t \,|\, s_t = s] \\
&= \mathbb{E}[r_{t+1} + \gamma r_{t+2} + \gamma^2 r_{t+3} + \cdots \,|\, s_t = s] \\
&= \mathbb{E}[r_{t+1} + \gamma G_{t+1} \,|\, s_t = s] \\
&= \mathbb{E}[r_{t+1} \,|\, s_t = s] + \gamma \mathbb{E}[G_{t+1} \,|\, s_t = s] \\
&= R_s + \gamma \mathbb{E}_{s' \sim P}\Big[\mathbb{E}[G_{t+1} \,|\, s_t = s, s_{t+1} = s']\Big] \\
&= R_s + \gamma \sum_{s'} P_{ss'} V(s')
\end{aligned}
$$

Bellman Expectation Equation for an MRP

$$V(s) = R_s + \gamma \sum_{s'} P_{ss'} V(s')$$

Matrix form:

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

Question: Why is the recursive Bellman equation reasonable?

How to Solve the Bellman Expectation Equation?

Matrix form:

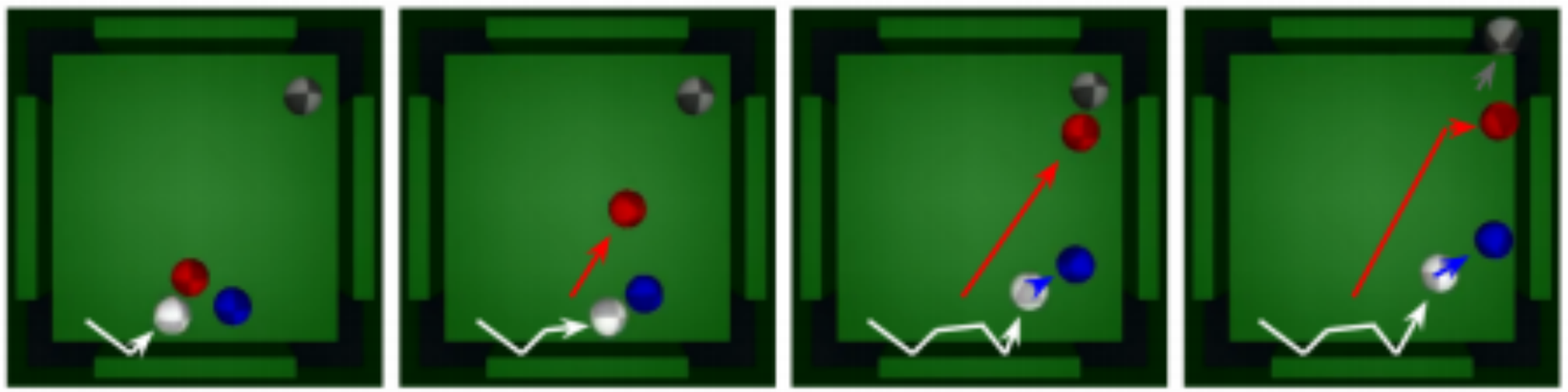
$$V = R + \gamma PV \quad \Rightarrow \quad V = (I - \gamma P)^{-1} R$$

- Issue: directly solvable only for small n (as the complexity of matrix inversion is typically $O(n^3)$)

- We will come back to this issue momentarily

Application of MRP Formulation: Predictron

- MRP can be a useful model for prediction tasks (i.e., no control)
- Example: *Predictron*



- Learn to predict future events for each ball, given 5 RGB frames as input
- Each event occurrence provides +1 reward

(Events: collision with balls, entering a packet …etc)

[Silver, ICML 2017]

Discount Factor

- Question: Why discount factor γ?
  1. Mathematically:
     - For the convergence issue
     - Avoids infinite returns in cyclic processes
  2. Philosophically:
     - Tradeoff between immediate rewards vs future rewards

- Typical choices of γ
  - Continuing environment: fixed $\gamma < 1$ (e.g. $\gamma = 0.9$)
  - Episodic environment: $\gamma \leq 1$

Beyond a Constant γ: Tuning γ on-the-fly via *Meta Gradient*

Z. Xu, H. van Hasselt, and D. Silver, "Meta-Gradient Reinforcement Learning" (NeurIPS 2018)

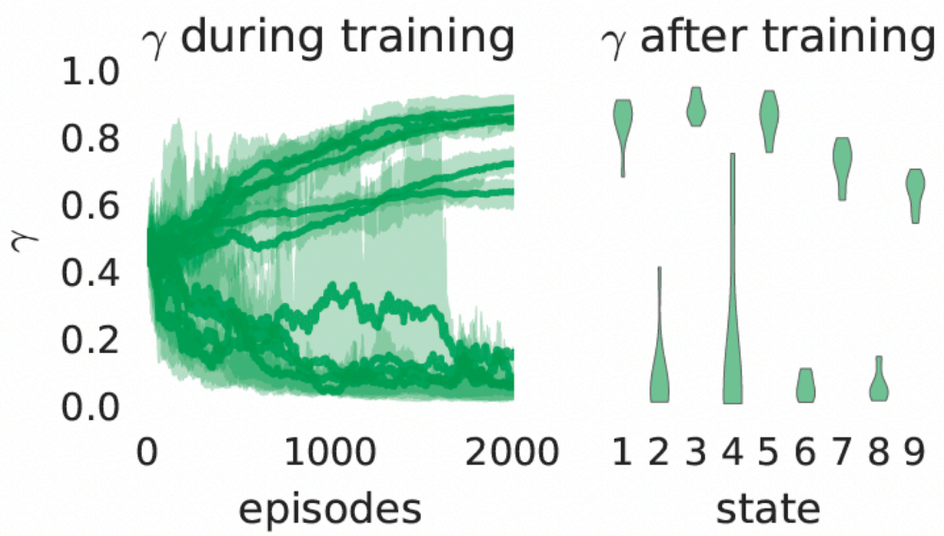
- Treat γ as a meta-parameter (to be tuned online via online cross validation)

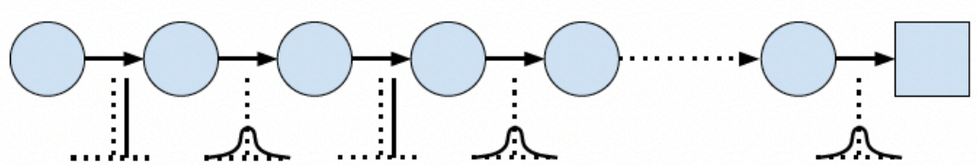
given by an RL algorithm

Model update: $\theta' = \theta + f(\theta, \gamma; \mathcal{D}_1)$ ← 1st batch of samples

objective (e.g., total return) ↘ ; 2nd batch of samples ↙

γ update: $\gamma' = \gamma + \alpha \cdot \dfrac{\partial J(\theta', \gamma; \mathcal{D}_2)}{\partial \gamma}$





(a) Rewards alternate between $+0.1$ or randomly sampling from a zero-mean Gaussian.

What if we have some control over state transitions?

Markov Decision Process (Formally)

- **Markov Decision Process (MDP)**: An MDP $(\mathcal{S}, \mathcal{A}, P, R, \gamma)$ is specified by

Underlying Dynamics

1. State space $\mathcal{S}$ (assumed finite)
2. Action space $\mathcal{A}$ (assumed finite)
3. Transition matrix $P = [P^a_{ss'}]$ with $P^a_{ss'} = \mathbb{P}[s_{t+1} = s' \,|\, s_t = s, a_t = a]$

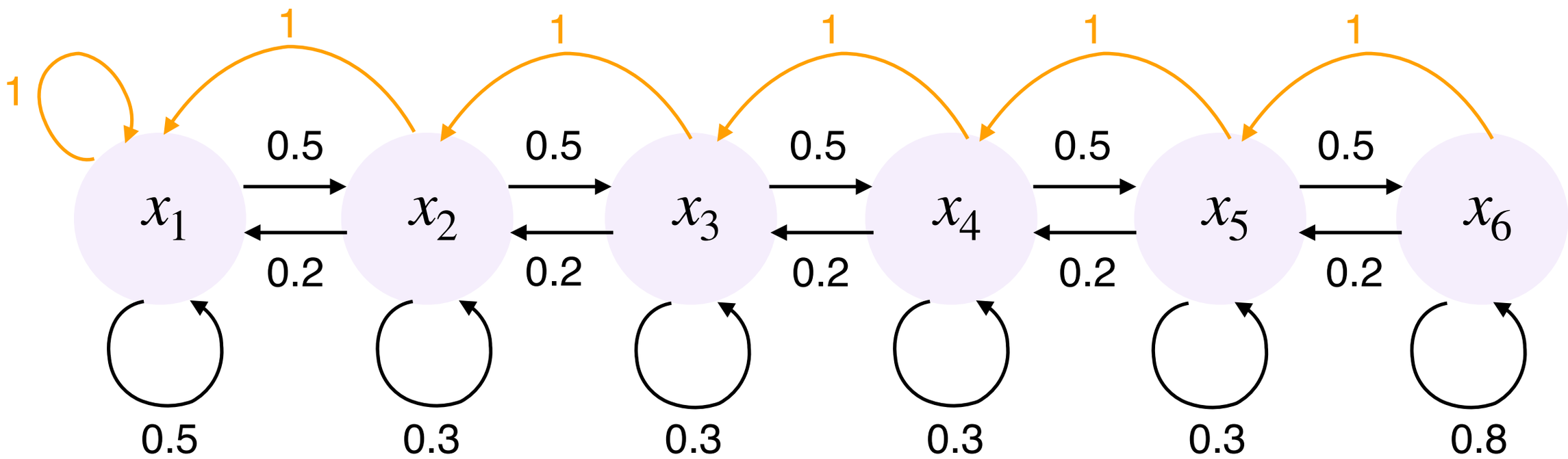
Task / Goal

4. Reward function $R^a_s = \mathbb{E}[r_{t+1} \,|\, S_t = s, A_t = a]$
5. Discount factor $\gamma \in [0,1]$

- In this lecture, we shall assume the model parameters P and R^a_s are known (i.e. no learning)

Example: N-Chain / Mars-Rover MDP

- Example: N-Chain with 2 actions (L & R) [ICML 2000, Strens]
  - ⟶ : transitions induced by R
  - ⟶ : transitions induced by L



- Reward = 0.05 at x_1, reward = 1 at x_6, and 0 elsewhere
- A sample trajectory is denoted by $s_0, a_0, r_1, s_1, a_1, r_2, \cdots$
  - e.g. x_2, L, 0, x_1, R, 0.05, x_2, R, 0, x_3…

How to Specify a Policy?

- Idea: “policy" is a lookup table specifying the action taken at any given state

- (Randomized) Policy: A policy π is a conditional distribution over possible actions given state s, i.e for any $s \in \mathcal{S}, a \in \mathcal{A}$

$$\pi(a \mid s) := \mathbb{P}(A_t = a \mid S_t = s)$$

- Remark: Here we focus on stationary policies, i.e. π does not depend on time t

[Puterman, 1994]

- Question: What’s the intuition behind using stationary policies?

Connection Between MDP and MRP

- Idea: Fix a policy $\pi(a \mid s)$ for an MDP $(\mathcal{S}, \mathcal{A}, P, R, \gamma)$:

1. What is the probability of $s \to s'$ under π?

$$P^{\pi}_{ss'} = \sum_{a \in \mathcal{A}} \pi(a \mid s) P^{a}_{ss'}$$

2. What is the expected reward of begin in s under π?

$$R^{\pi}_{s} = \sum_{a \in \mathcal{A}} \pi(a \mid s) R^{a}_{s}$$

- Under a fixed $\pi(a \mid s)$, we get an π-induced MRP $(\mathcal{S}, P^{\pi}, R^{\pi}, \gamma)$

Goals, Return, and State-Value Function of MDPs

- Goal: Given P and R, find a policy π that maximizes the expected cumulative reward (Q: this formulation can be viewed as _____?)

- Return G_t: Cumulative discounted rewards over a single trajectory from t onwards (random)

$$G_t = r_t + \gamma r_{t+1} + \gamma^2 r_{t+2} + \cdots = \sum_{k=0}^{\infty} \gamma^k r(s_{t+k}, a_{t+k})$$

- State-value function $V^\pi(s)$: Expected return if we start from state s

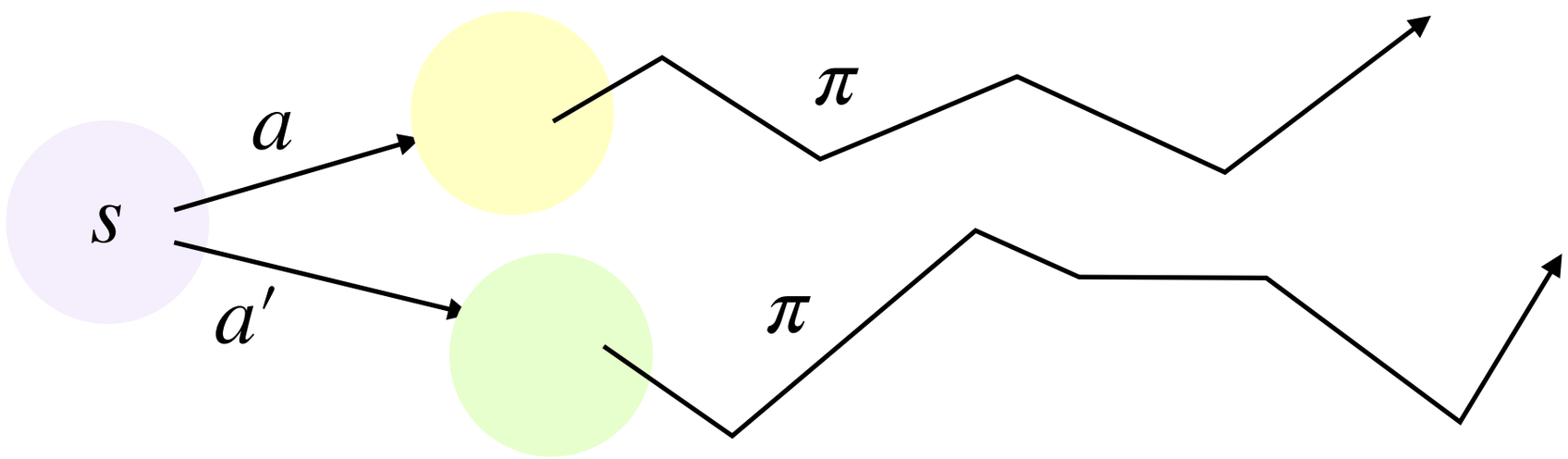
$$V^\pi(s) = \mathbb{E}[G_t \,|\, s_t = s; \pi]$$

- Question: The expectation above is taken w.r.t. randomness of ?

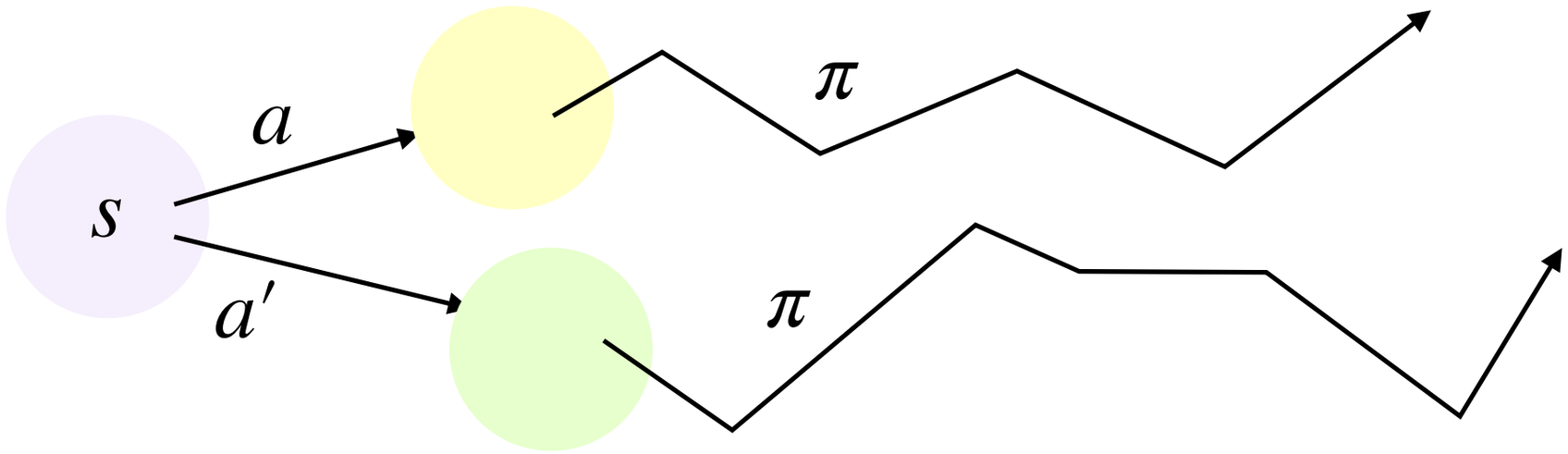
Action-Value Function $Q^{\pi}(s, a)$

- Action-value function $Q^{\pi}(s, a)$: Expected return if we start from state s and take action a, and then follow policy π

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



Natural Connection Between $V^{\pi}(s)$ and $Q^{\pi}(s,a)$



(1) V written in Q

$$V^{\pi}(s) = \sum_{a \in \mathcal{A}} \pi(a \mid s) Q^{\pi}(s,a)$$

(2) Q written in V

$$Q^{\pi}(s,a) = R_s^a + \gamma \sum_{s' \in \mathcal{S}} P_{ss'}^a V^{\pi}(s')$$

Recursions for Computing $V^{\pi}(s)$ and $Q^{\pi}(s,a)$

(1) V written in Q

$$V^{\pi}(s) = \sum_{a \in \mathcal{A}} \pi(a \mid s) Q^{\pi}(s,a)$$

(2) Q written in V

$$Q^{\pi}(s,a) = R_s^a + \gamma \sum_{s' \in \mathcal{S}} P_{ss'}^a V^{\pi}(s')$$

(3) V written in V

(4) Q written in Q

(Non-Iterative) MDP Policy Evaluation

For $V^\pi(s)$:

$$V^\pi(s) = \sum_{a \in \mathcal{A}} \pi(a \mid s) \left(R_s^a + \gamma \sum_{s' \in \mathcal{S}} P_{ss'}^a V^\pi(s') \right)$$

Consider π-induced MRP $(\mathcal{S}, P^\pi, R^\pi, \gamma)$:

$$R_s^\pi = \sum_{a \in \mathcal{A}} \pi(a \mid s) R_s^a$$

$$P_{ss'}^\pi = \sum_{a \in \mathcal{A}} \pi(a \mid s) P_{ss'}^a$$

Matrix form:

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

Solution of V^π:

Iterative MDP Policy Evaluation (IPE)

We know:

$$V^{\pi}(s) = \sum_{a \in \mathcal{A}} \pi(a \mid s) \left(R_s^a + \gamma \sum_{s' \in \mathcal{S}} P_{ss'}^a V^{\pi}(s') \right)$$

- Iterative policy evaluation for a fixed policy π:

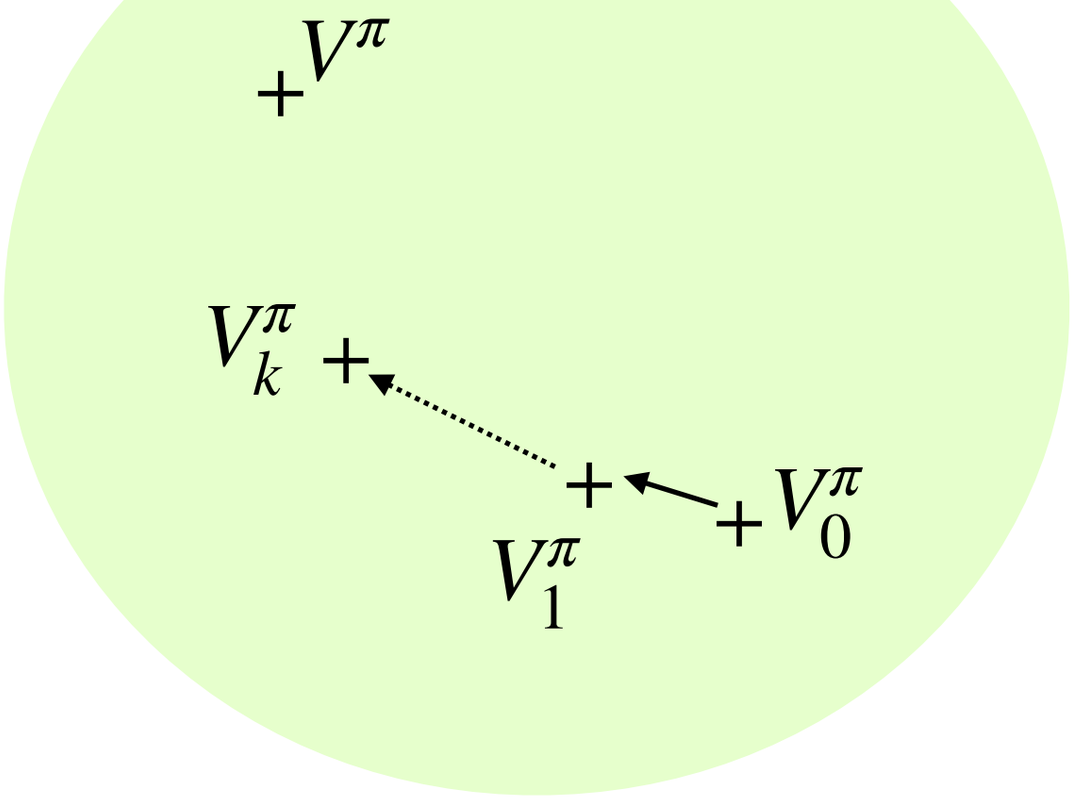
  1. Initialize $V_0^{\pi}(s) = 0$ for all s

  2. For $k = 1,2,...$

$$V_k^{\pi}(s) = \sum_{a \in \mathcal{A}} \pi(a \mid s) \left(R_s^a + \gamma \sum_{s' \in \mathcal{S}} P_{ss'}^a V_{k-1}^{\pi}(s') \right) \quad \text{for all } s$$

- Question: What if we start from $V_0^{\pi}(s) = V^{\pi}(s), \forall s$?
- Question: In general, does $V_k^{\pi}(s)$ converge to the correct $V^{\pi}(s)$?

(Complete) Metric vector space $\mathbb{R}^{|\mathcal{S}|}$



- Question: What does IPE do to points in this space?

Prove convergence in 2 steps:

(A1): IPE brings points closer (formally, a contraction operator)

(A2): Under any contraction operator, the points converges to a unique fixed point

For (A1): IPE is a Contraction Map

- IPE operator (aka Bellman expectation backup operator):

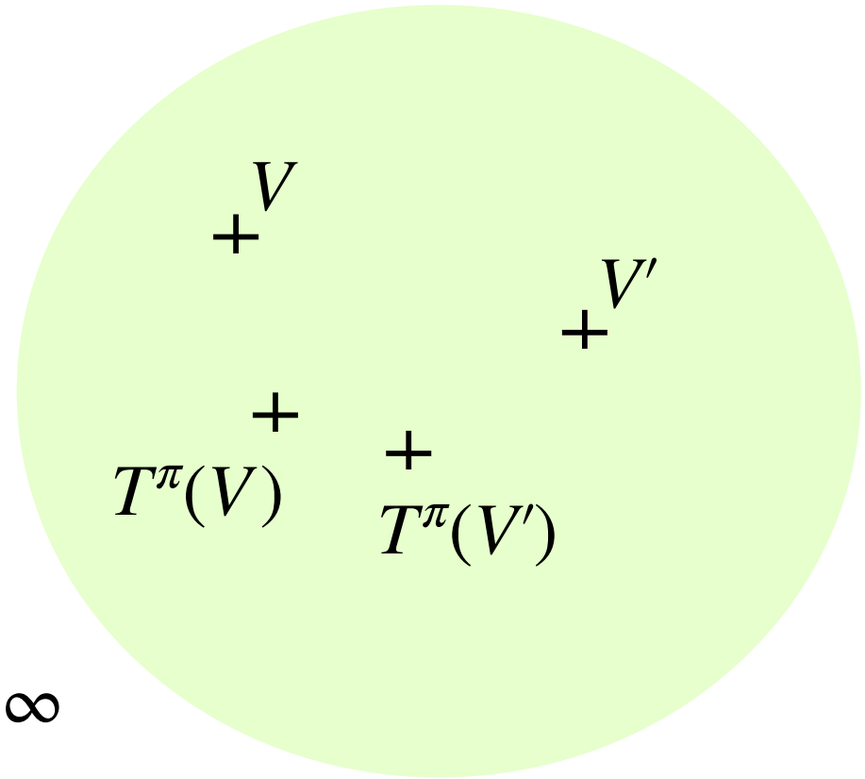
$$T^{\pi}(V) := R^{\pi} + \gamma P^{\pi} V$$

- Consider $\underline{L_{\infty}\text{-norm}}$ to measure distance between any two value functions V, V'

$$||V - V'||_{\infty} := \max_{s \in \mathcal{S}} |\, V(s) - V'(s) \,|$$

For (A1): IPE is a Contraction Map (Cont.)

- IPE operator: $T^\pi(V) = R^\pi + \gamma P^\pi V$



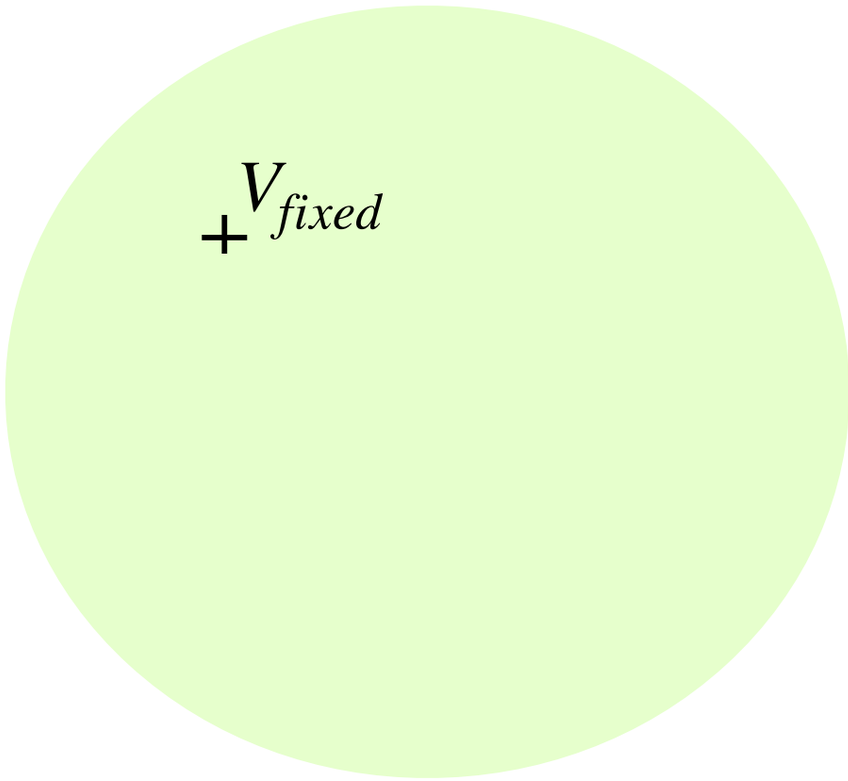
$$
\begin{aligned}
&||T^\pi(V) - T^\pi(V')||_\infty \\
&= ||(R^\pi + \gamma P^\pi V) - (R^\pi + \gamma P^\pi V')||_\infty \\
&= \gamma ||P^\pi (V - V')||_\infty \\
&\leq \gamma ||(V - V')||_\infty
\end{aligned}
$$

We say T^π is a γ-contraction operator ($\gamma < 1$)

For (A2): Banach Fixed-Point Theorem

- **Banach Fixed-Point Theorem:** For any non-empty complete metric space, if T is a γ-contraction operator, then T has a unique fixed point.

- Question: Why is this useful?



Quick Summary

- Under IPE, the value functions V_k^π converges to the correct V_k^π, for any initialization V_0^π

Contraction property is central to various RL algorithms:

A Theory of Regularized Markov Decision Processes

Matthieu Geist [1] Bruno Scherrer [2] Olivier Pietquin [1]

Abstract

Many recent successful (deep) reinforcement learning algorithms make use of regularization, generally based on entropy or Kullback-Leibler divergence. We propose a general theory of regularized Markov Decision Processes that generalizes these approaches in two directions: we consider a larger class of regularizers, and we consider the general modified policy iteration approach, encompassing both policy iteration and value iteration. The core building blocks of this theory are a notion of regularized Bellman operator and the Legendre-Fenchel transform, a classical tool of convex optimization. This approach allows for error propagation analyses of general algorithmic schemes of which (possibly variants of) classical algorithms such as Trust Region Policy Optimization, Soft Q-learning, Stochastic Actor Critic or Dynamic Policy Programming are special cases. This also draws connections to proximal convex optimization, especially to Mirror Descent.

Tsallis entropy (Lee et al., 2018 having a sparse regularized greedy are based on a notion of tempo somehow extending the notion o regularized case (Nachum et al. Nachum et al., 2018), or on policy Mnih et al., 2016).

This non-exhaustive set of algori ing regularization, but they are different principles, consider eac ization, and have ad-hoc analysis, a general theory of regularized M (MDPs). To do so, a key observa dynamic programming, or (A)D from the core definition of the B tor. The framework we propose i Bellman operator, and on an assc transform. We study the theoretic larized MDPs and of the related r This generalizes many existing th vides new ones. Notably, it allow

(Geist et al., ICML 2019)

A Generalized Algorithm for Multi-Objective Reinforcement Learning and Policy Adaptation

Runzhe Yang
Department of Computer Science
Princeton University
runzhey@cs.princeton.edu

Xingyuan Sun
Department of Computer Science
Princeton University
xs5@cs.princeton.edu

Karthik Narasimhan
Department of Computer Science
Princeton University
karthikn@cs.princeton.edu

Abstract

We introduce a new algorithm for multi-objective reinforcement learning (MORL) with linear preferences, with the goal of enabling few-shot adaptation to new tasks. In MORL, the aim is to learn policies over multiple competing objectives whose relative importance (*preferences*) is unknown to the agent. While this alleviates dependence on scalar reward design, the expected return of a policy can change significantly with varying preferences, making it challenging to learn a single model to produce optimal policies under different preference conditions. We propose a generalized version of the Bellman equation to learn a single parametric representation for optimal policies over the space of all possible preferences. After an initial learning phase, our agent can execute the optimal policy under any given preference, or automatically infer an underlying preference with very few samples. Experiments across four different domains demonstrate the effectiveness of our approach.[1]

(Yang et al., NeurIPS 2019)