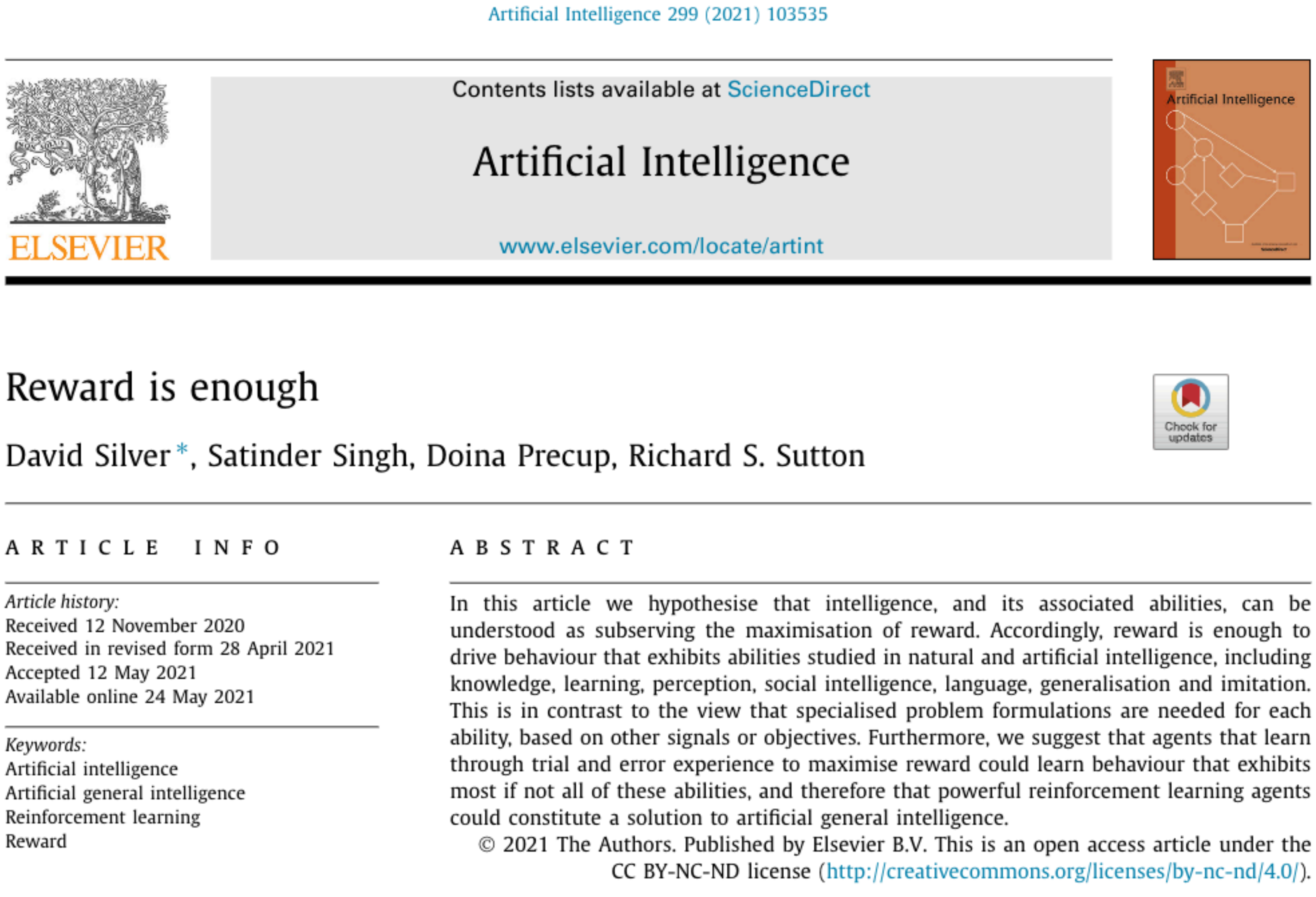
535514: Reinforcement Learning
Lecture 2 — Markov Decision Process

Ping-Chun Hsieh

February 22, 2024

Reward is Enough?



Artificial Intelligence 299 (2021) 103535

Contents lists available at ScienceDirect

Artificial Intelligence

www.elsevier.com/locate/artint

ELSEVIER

Reward is enough

David Silver *, Satinder Singh, Doina Precup, Richard S. Sutton

Check for updates

ARTICLE INFO

Article history:
Received 12 November 2020
Received in revised form 28 April 2021
Accepted 12 May 2021
Available online 24 May 2021

Keywords:
Artificial intelligence
Artificial general intelligence
Reinforcement learning
Reward

ABSTRACT

In this article we hypothesise that intelligence, and its associated abilities, can be understood as subserving the maximisation of reward. Accordingly, 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. This is in contrast to the view that specialised problem formulations are needed for each ability, based on other signals or objectives. Furthermore, we suggest that agents that learn through trial and error experience to maximise reward could learn behaviour that exhibits most if not all of these abilities, and therefore that powerful reinforcement learning agents could constitute a solution to artificial general intelligence.

David Silver

Satinder Singh

Doina Precup

Richard Sutton

"… Accordingly, reward is enough to drive behavior that exhibits abilities studied in natural and artificial intelligence, including knowledge, learning, perception, social intelligence, language, generalization and imitation"

Reward Design is Certainly Challenging!

"…We assumed the score the player earned would reflect the informal goal of finishing the race, so we included the game in an internal benchmark designed to measure the performance of reinforcement learning systems on racing games.
However, it turned out that the targets were laid out in such a way that the reinforcement learning agent could gain a high score without having to finish the course. This led to some unexpected behavior when we trained an RL agent to play the game."

https://openai.com/blog/faulty-reward-functions/

Review: Markov Reward Process

- **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-to-go)

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

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

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

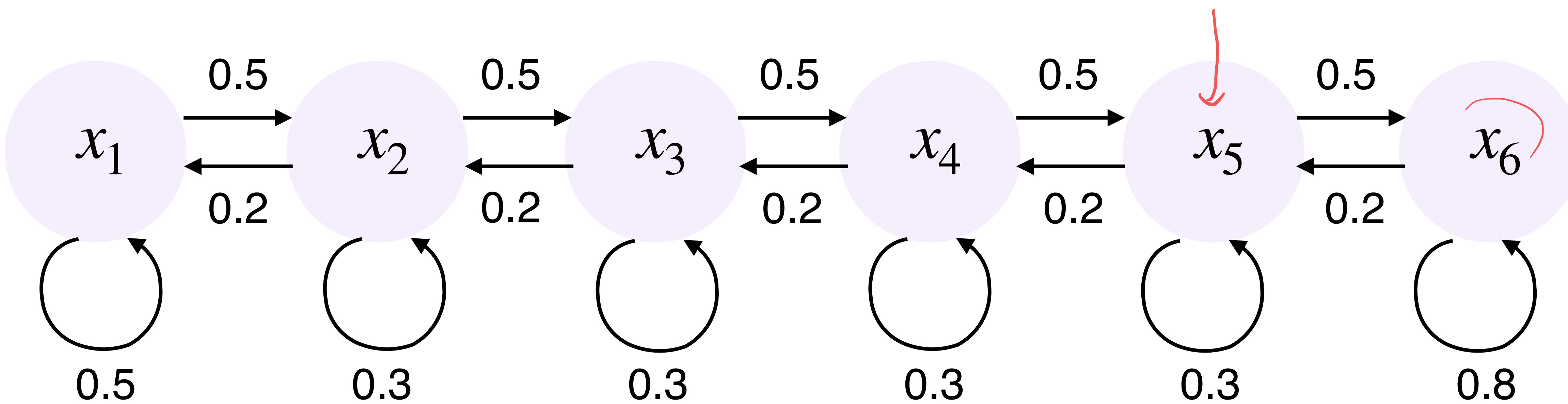
$$V(s) = \mathbb{E}[G_t \,|\, s_t = s]$$

- 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

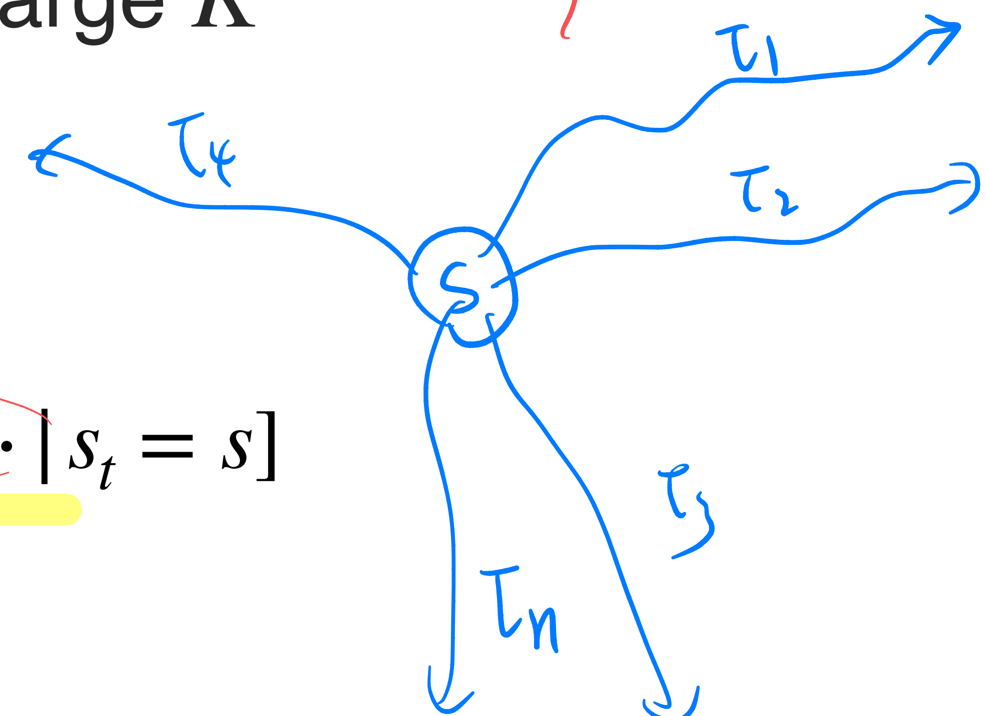
- Sample return for a 5-step episode x_5, x_6, x_6, x_5, x_4 with $\gamma = 0.9$
  - $G_t = 0 + (1 \times 0.9) + (1 \times 0.9^2) + (0 \times 0.9^3) + (0 \times 0.9^4)$

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

Given P, R, how to find $V(s)$, for every $s \in S$?

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

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



$V(s) \approx \frac{1}{n} \sum_{i=1}^{n} G_t(\tau_i)$

return obtained along trajectory τ_i

2. Recursion: Use dynamic programming

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

Law of iterated expectation (LIE)

Monte-Carlo estimates

X is a random variable, $X \sim N(0,1)$

$E[f(X)]$ e.g. $f(x) = e^{x^3} \cdot \sin(\pi x) \cdot \log(|x|)$

$$= \int_{-\infty}^{+\infty} f(x) \cdot \frac{1}{\sqrt{2\pi}} e^{-\frac{x^2}{2}} \, dx$$

$X_1, X_2, \cdots, X_n \sim N(0,1)$

$$E[f(X)] \approx \frac{1}{n} \sum_{i=1}^{n} f(X_i) \quad \text{(by SLLN)}$$

Bellman Expectation Equation for an MRP

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

for every s

$|S| = 100$

$$(V - \gamma P V) = R$$

$$(I - \gamma P) V = R$$

Matrix form: $$V = R + \gamma P V$$

$P_{s^{(i)} s^{(j)}}$

$$\begin{bmatrix} V(s^{(1)}) \\ \vdots \\ V(s^{(100)}) \end{bmatrix} = \begin{bmatrix} R_{s^{(1)}} \\ \vdots \\ R_{s^{(100)}} \end{bmatrix} + \gamma \cdot \begin{bmatrix} P_{s^{(1)} s^{(1)}} & \cdots & P_{s^{(1)} s^{(100)}} \\ \vdots & \ddots & \vdots \\ P_{s^{(100)} s^{(1)}} & \cdots & P_{s^{(100)} s^{(100)}} \end{bmatrix} \begin{bmatrix} V(s^{(1)}) \\ \vdots \\ V(s^{(100)}) \end{bmatrix}$$

Question: Why is the recursive Bellman equation reasonable?

How to Solve the Bellman Expectation Equation?

$|S| = 10^{30}$ states

Matrix form:

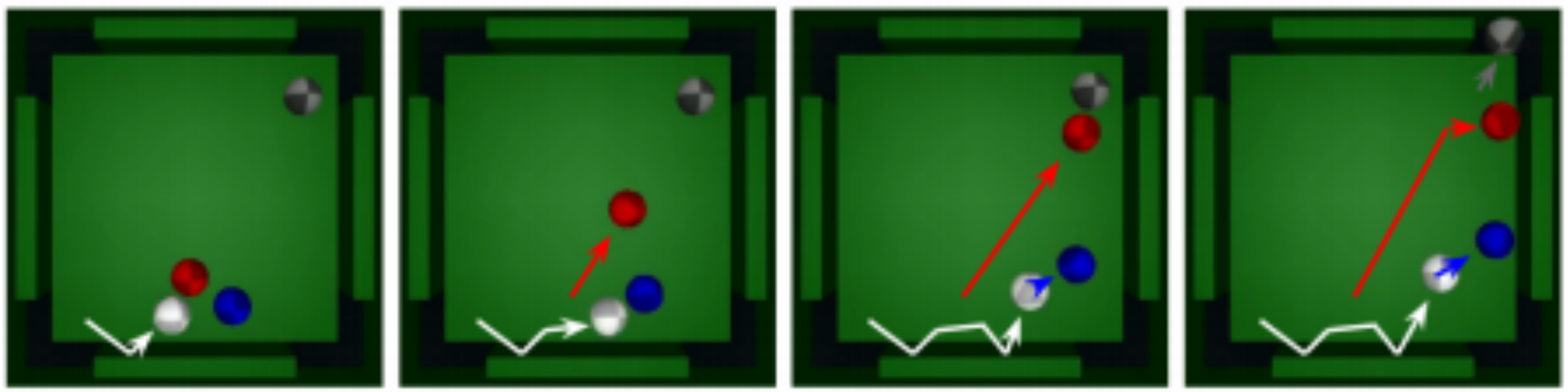
$$V = R + \gamma P V \quad \Rightarrow \quad V = (I - \gamma P)^{-1} R$$

- Issue: This is 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 <u>predict future events</u> 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: $\gamma \leq 1$

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 $P^a_{ss'} = \mathbb{P}[s_{t+1} = s' \mid s_t = s, a_t = a]$

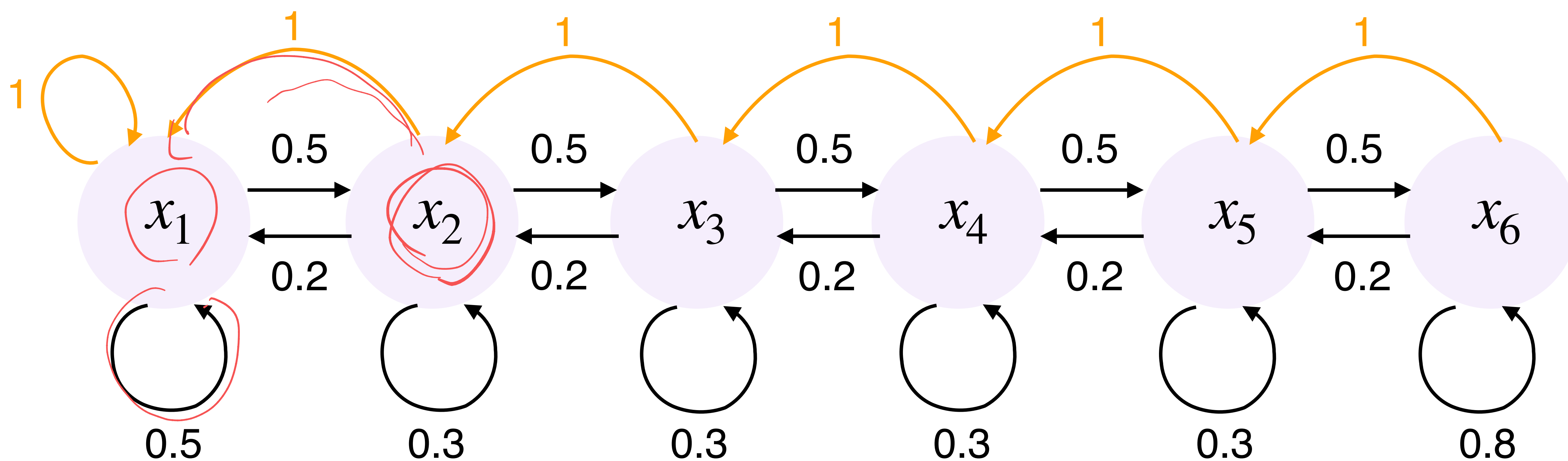
Task / Goal

4. Reward function $R_{s,a} = \mathbb{E}[r_{t+1} \mid s_t = s, a_t = a]$
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?

$|S|=4$, $|A|=5$

state \ a_i	$a^{(1)}$	$a^{(2)}$	$a^{(3)}$	$a^{(4)}$	$a^{(5)}$
$s^{(1)}$	0.2	0.1	0.15	0.35	0.2
$s^{(2)}$					
$s^{(3)}$					
$s^{(4)}$					

- **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 $\pi(a\,|\,s)$ for an MDP $(\mathcal{S}, \mathcal{A}, P, R, \gamma)$:

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

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

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

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

- Under a fixed $\pi(a\,|\,s)$, 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 (Question: this formulation can be viewed as optimal control, model-based RL, or model-free RL?)

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

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

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

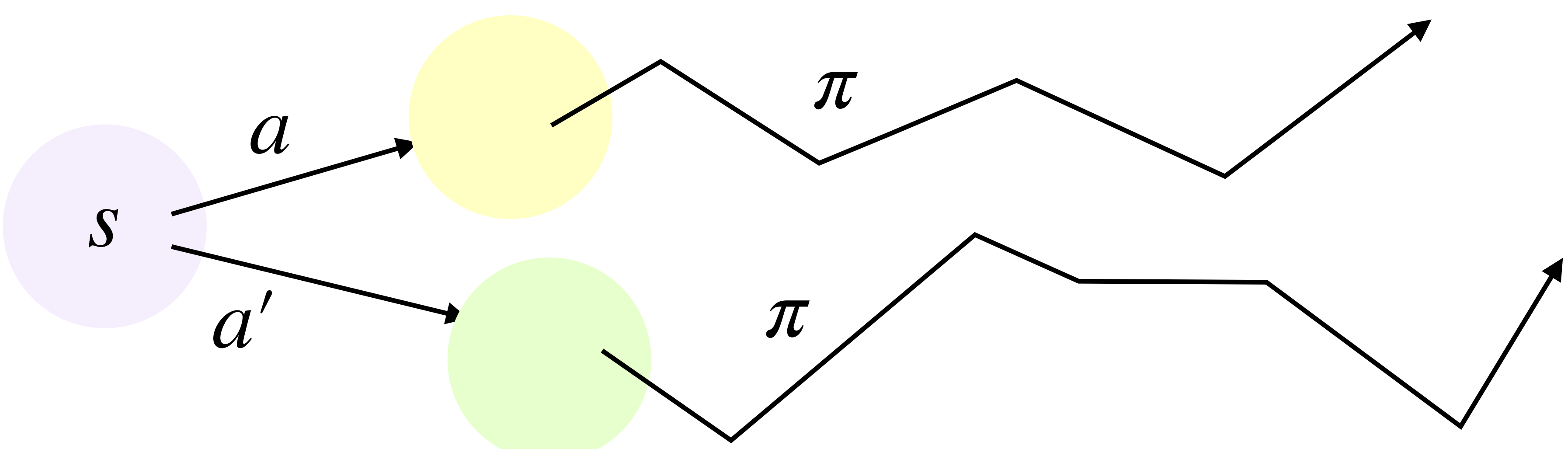
$$V^\pi(s) = \mathbb{E}[G_t \,|\, s_t = s; \pi]$$

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

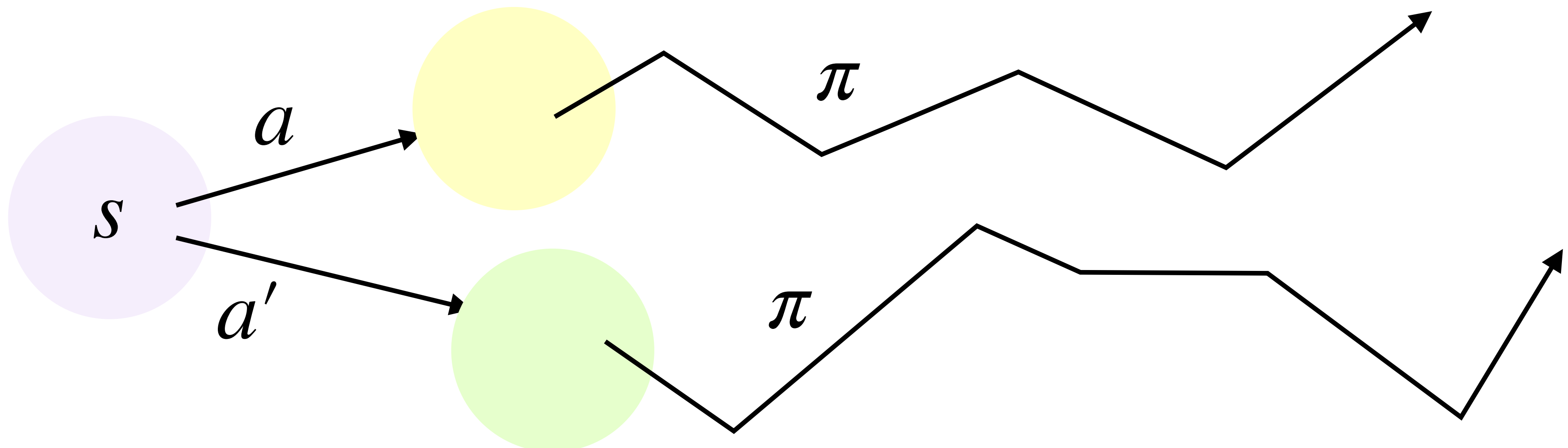
Action-Value Function $Q^{\pi}(s, a)$

- 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

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

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

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

(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: $$V^\pi(s) = \sum_{a \in \mathcal{A}} \pi(a \mid s)\Big(R_{s,a} + \gamma \sum_{s' \in \mathcal{S}} P^a_{ss'} V^\pi(s')\Big)$$

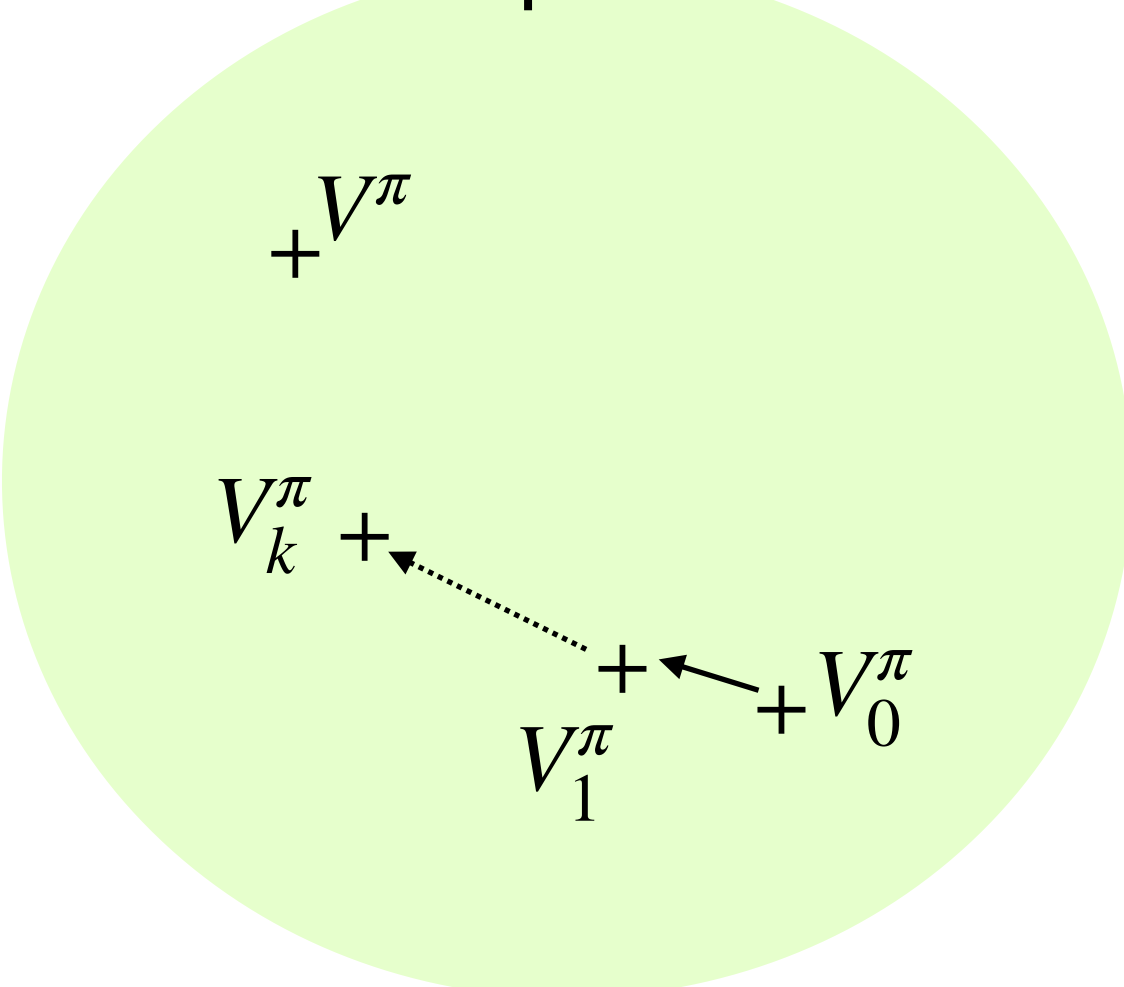
- Iterative policy evaluation for a fixed policy π:

  1. Initialize $V^\pi_0(s) = 0$ for all s
  2. For $k = 1,2,\ldots$

$$V^\pi_k(s) = \sum_{a \in \mathcal{A}} \pi(a \mid s)\Big(R_{s,a} + \gamma \sum_{s' \in \mathcal{S}} P^a_{ss'} V^\pi_{k-1}(s')\Big) \quad \text{for all } s$$

- Question: What if we start from $V^\pi_0(s) = V^\pi(s), \forall s$?
- Question: In general, does $V^\pi_k(s)$ 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 <u>contraction operator</u>)

(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):

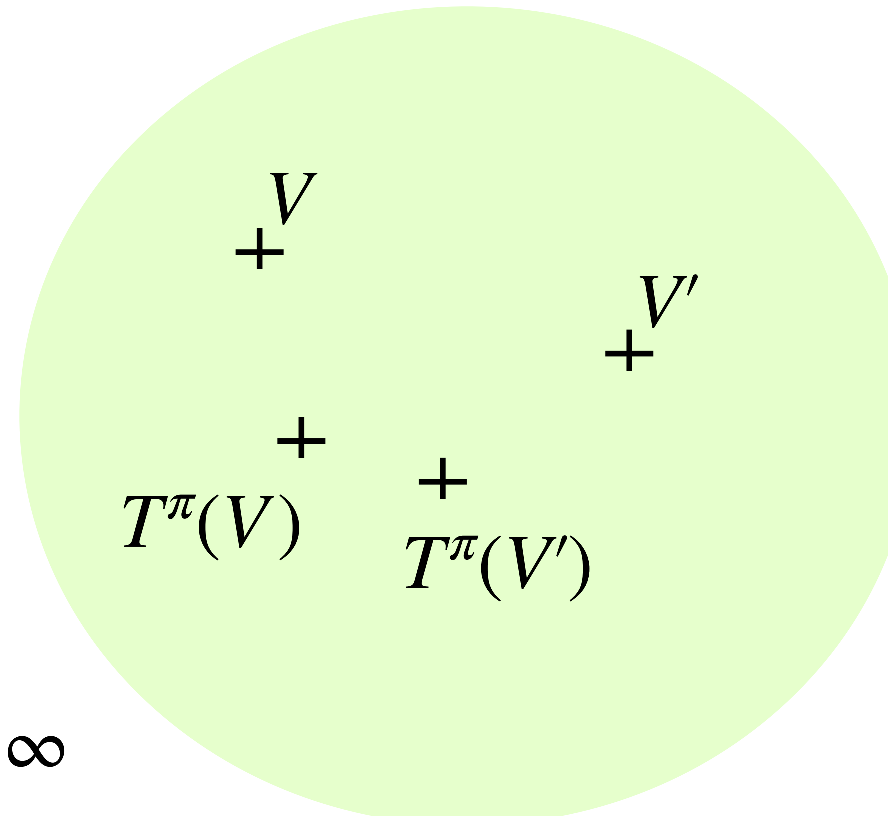
$$T^{\pi}(V) := R^{\pi} + \gamma P^{\pi} V$$

- Consider L_∞-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: $T^\pi(V) = R^\pi + \gamma P^\pi V$



For any two value functions V and V',

$$||T^\pi(V) - T^\pi(V')||_\infty$$

$$= ||(R^\pi + \gamma P^\pi V) - (R^\pi + \gamma P^\pi V')||_\infty$$

$$= \gamma ||P^\pi(V - V')||_\infty$$

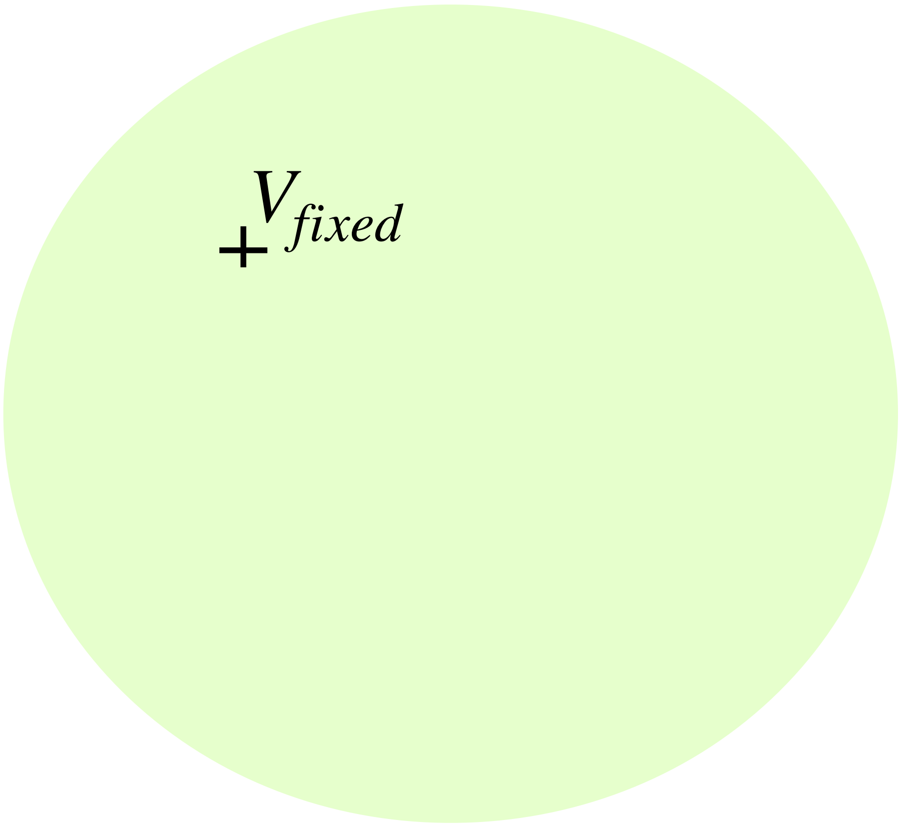
$$\leq \gamma ||(V - V')||_\infty$$

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 asso
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)