Successor Feature Sets: Generalizing Successor Representations Across Policies

Kianté Brantley,[1] Soroush Mehri,[2] and Geoff Gordon[2]

[1] University of Maryland, [2] Microsoft Research

Motivation:

In a RL problem, there are three important things that we can change

- we can change the **state** of the agent
- we can change the **policy**
- we can change the task or **reward function**

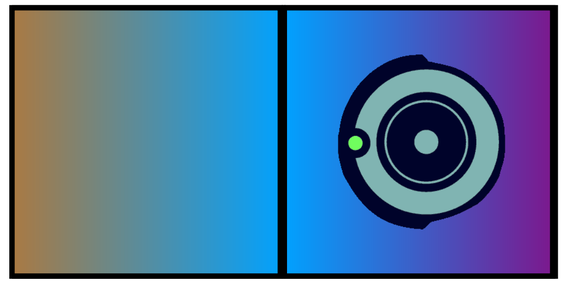
If we change any of these, we would like to generalize

In this talk we ask, can we handle all three at once?

Partially Observable Markov Decision Process

A partially observable Markov decision process (POMDP) can be defined as a tuple ⟨S, A, T , R, Ω, O⟩, where:

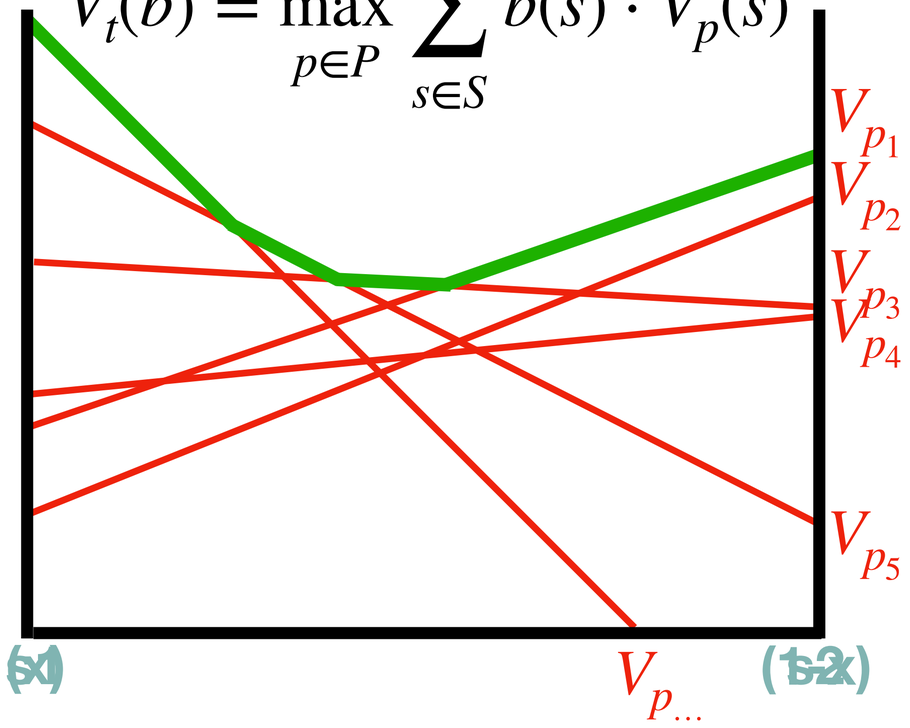
- S - **set of states** **[two states: s1 s2]**
- A - **set of actions** **[two actions: a1 (left) a2 (right)]**
- T - S × A → Π (S) **is the state-transition function**
- R - S × A→R **is the reward function**
- Ω - **set of observations** **[three observations: z1, z2, z3]**
- O - S × A → Π(Ω) **is the observation function**
- b - **belief state that summarizes previous experiences**



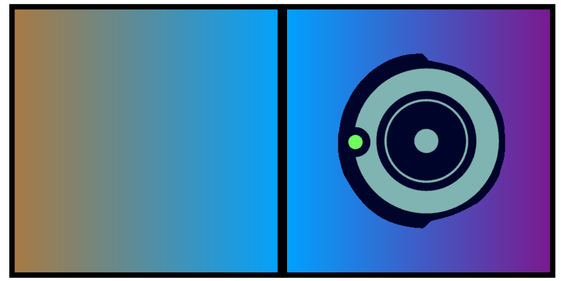
POMDP Value Iteration

Let P be the finite set of all t-step policy trees. Then

$$V_t(b) = \max_{p \in P} \sum_{s \in S} b(s) \cdot V_p(s)$$



1D belief space for a 2 state POMDP



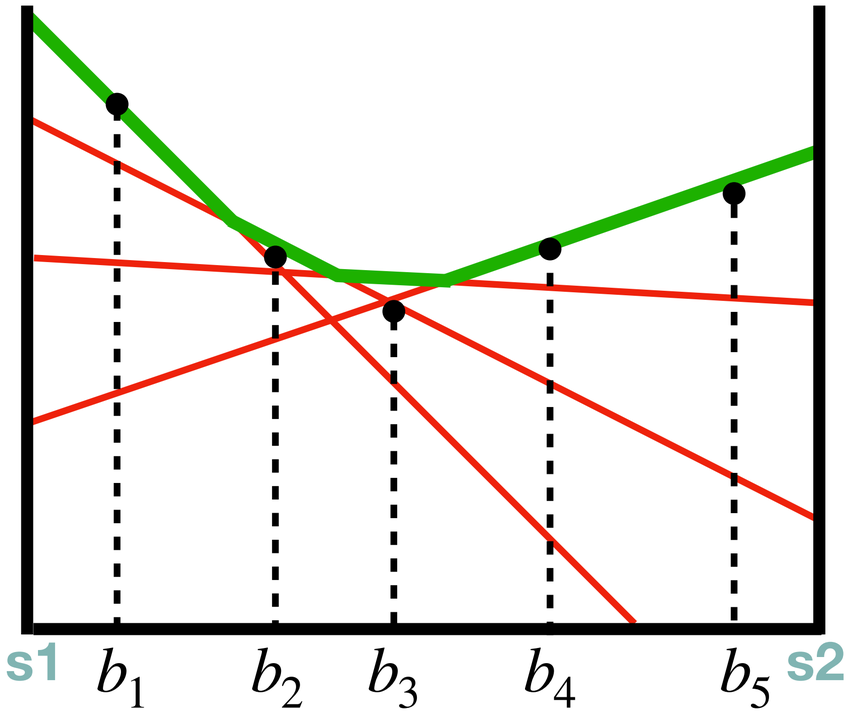
Planning and acting in partially observable stochastic domains Leslie Pack Kaelbling, Michael L. Littman, Anthony R. Cassandra 1998

Tony's POMDP Page - Brown University Computer Science

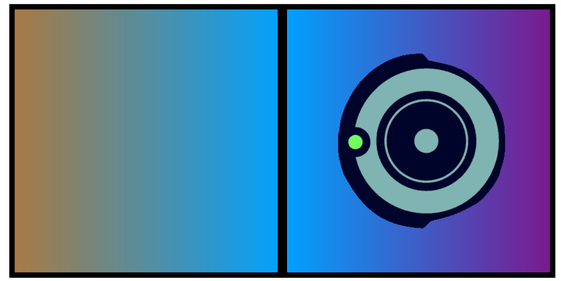
POMDP Value Iteration

(point-based value iteration)

Selects small set of representative belief points b_k



1D belief space for a 2 state POMDP

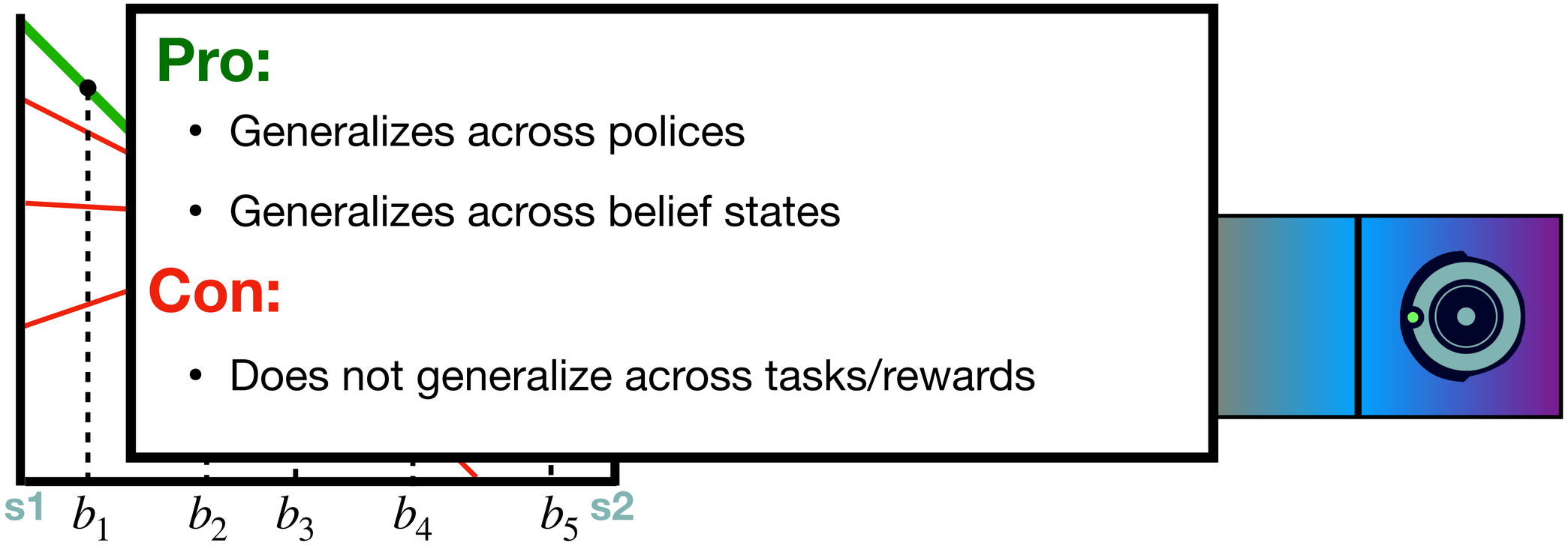


Point-based value iteration: An anytime algorithm for POMDPs Joelle Pineau, Geoff Gordon and Sebastian Thrun

POMDP Value Iteration

(point-based value iteration)

Selects small set of representative belief points b_k



1D belief space for a 2 state POMDP

Successor Representation

(reward function is decoupled from transitions)

- Successor representation learns two quantities
    1. Immediate reward

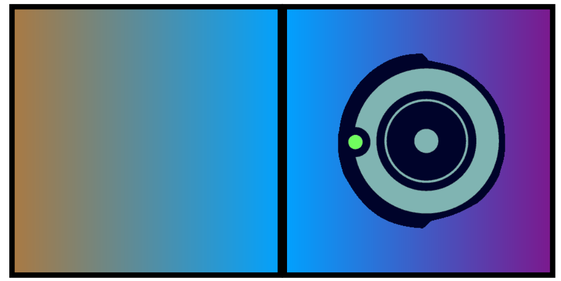
$$r(s) = \mathbb{E}_{\pi}[r_{t+1} \,|\, s_t = s]$$

    2. Discount future state occupancy

$$M(s, s') = \mathbb{E}_{\pi}[\sum_{t=0}^{\infty} \gamma^t 1(s = s') \,|\, s_0 = s]$$

- Value function

$$V^{\pi}(s) = \sum_{s'} M(s, s') r(s')$$



Improving Generalization for Temporal Difference Learning: The Successor Representation Dayan 1993

Successor Representation

(reward function is decoupled from transitions)

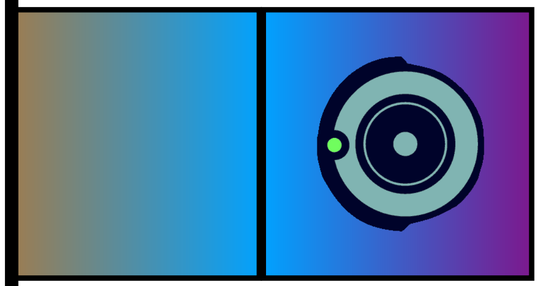
- Successor representation learns two quantities
  1. Im
  2. Di
- Value f

Pro:

- Generalizes across tasks/rewards
- Generalize across states
- Reward is linear in the successor representation matrix

Con:

- Limited generalization across polices, e.g. via GPI
- The state space in our problems are huge so we need to generalize the notion of successor representation



Successor Features

(extending successor representations to features)

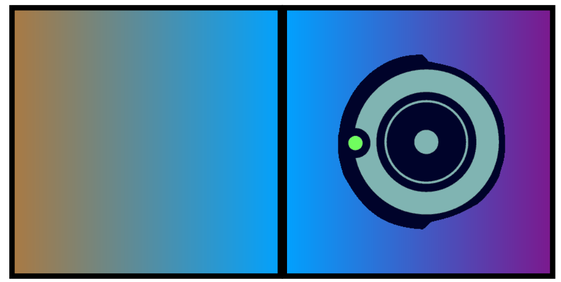
- Successor Features learns two quantities

1. Immediate reward

$$r(s, a, s') = \phi(s, a, s')^T w \quad \textbf{where} \quad \phi(s, a, s') \in \mathbb{R}^d \textbf{ are features}$$

2. Successor Features

$$\psi^{\pi}(s, a) = \mathbb{E}^{\pi}\left[\sum_{i=t}^{\infty} \gamma^{i-t} \phi_{i+1} \mid S_t = s, A_t = a\right]$$



Action-Value Function:

$$Q^{\pi}(s, a) = \psi^{\pi}(s, a)^T w$$

Successor Features for Transfer in Reinforcement Learning André Barreto, Will Dabney, Rémi Munos, Jonathan J. Hunt, Tom Schaul, Hado van Hasselt, David Silver

Universal Successor Features for Transfer Reinforcement Learning Chen Ma, Dylan R. Ashley, Junfeng Wen, Yoshua Bengio

Successor Features Combine Elements of Model-Free and Model-based Reinforcement Learning Lucas Lehnert, Michael L. Littman

Successor Features

(extending successor representations to features)

- Successor Features learns two quantities

1. Imme

$r(s, a$

2. Expec

$\psi^{\pi}(s,$

Actio

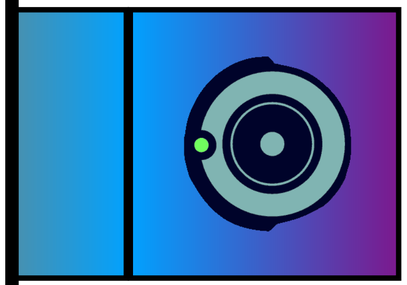
$Q^{\pi}(s,$

Pro:

- Generalizes across tasks/rewards
- Generalizes across states
- Extends successors representation beyond the tabular setting

Con:

- Limited generalization across polices, e.g., via GPI



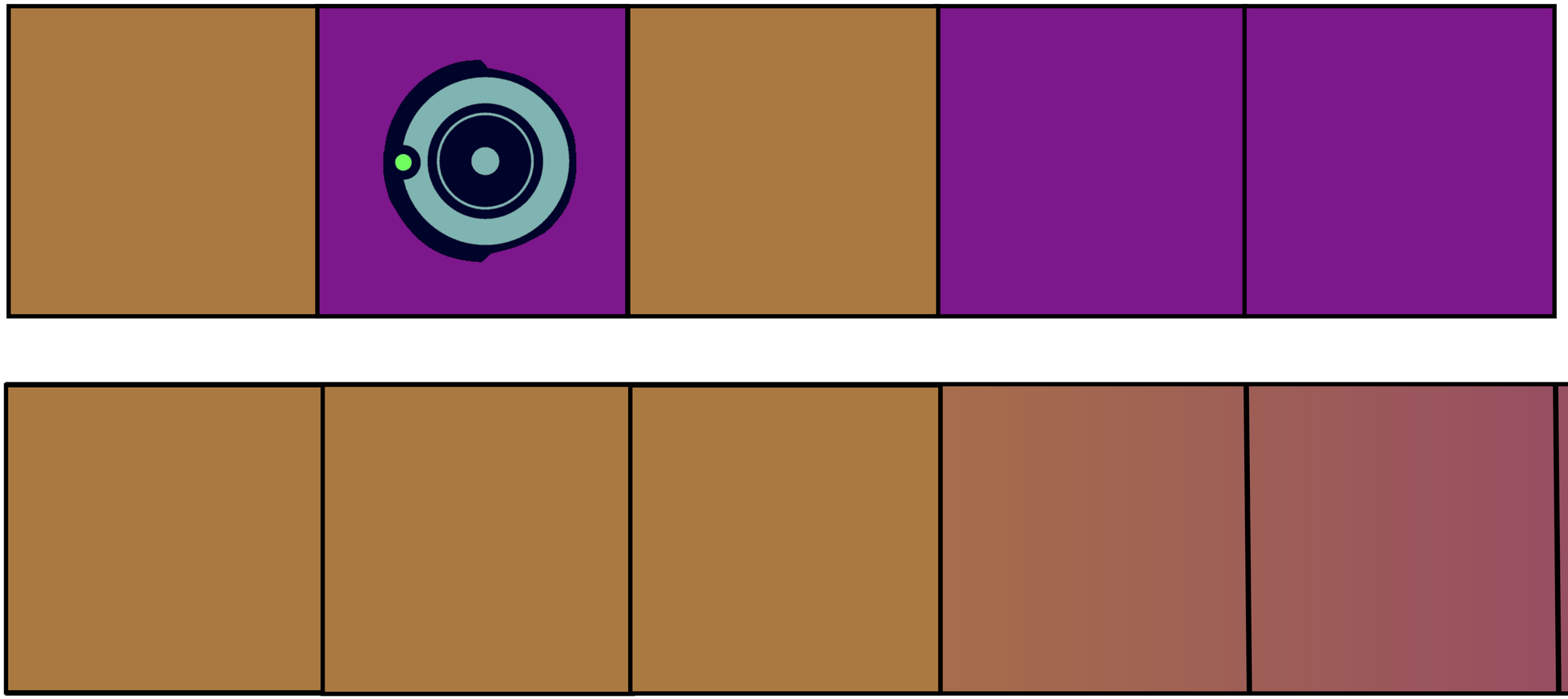
Successor Features for Transfer in Reinforcement Learning André Barreto, Will Dabney, Rémi Munos, Jonathan J. Hunt, Tom Schaul, Hado van Hasselt, David Silver

Universal Successor Features for Transfer Reinforcement Learning Chen Ma, Dylan R. Ashley, Junfeng Wen, Yoshua Bengio

Successor Features Combine Elements of Model-Free and Model-based Reinforcement Learning Lucas Lehnert, Michael L. Littman

Successor Features

(pictorial example)



Successor Features for Transfer in Reinforcement Learning André Barreto, Will Dabney, Rémi Munos, Jonathan J. Hunt, Tom Schaul, Hado van Hasselt, David Silver

Universal Successor Features for Transfer Reinforcement Learning Chen Ma, Dylan R. Ashley, Junfeng Wen, Yoshua Bengio

Successor Features Combine Elements of Model-Free and Model-based Reinforcement Learning Lucas Lehnert, Michael L. Littman

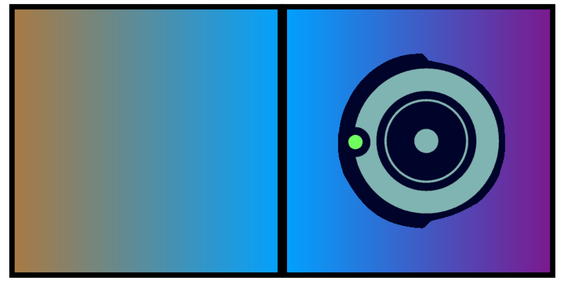
Predictive State Representations

(generalize POMDPs)

Changes the interpretation of the belief state:

Belief states no longer represent probability distribution over states, but instead satisfies two properties

Given starting state q_1, transition matrices T_{ao}, and a vector $u \in \mathbb{R}^k$



The two properties are:

1. $(\forall a, o, t) u^T T_{ao} q_t \geq 0$
2. $(\forall a, t) \sum_o u^T T_{ao} q_t = 1$

Predictive State Representations

(generalize POMDPs)

Changes the interpretation of the belief state:

Belie
but i

Give
a ve

The

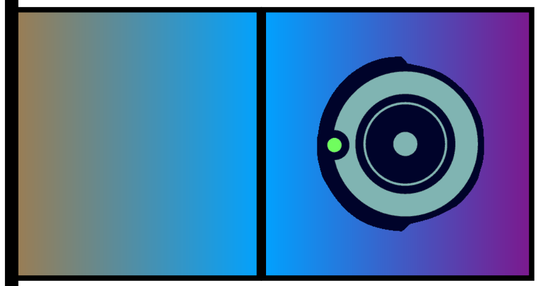
s,

Pro:

- Generalizes across states
- Generalize across polices
- Transitions and rewards are linear in the state vector q_t and the value is piecewise linear

Con:

- Does not generalize across rewards
- Learning PSRs is challenging



Our Contributions: Successor Feature Sets

For any control problem, we can write down

- a convex set of possible state representations q,
- a convex set of reward function representations r, and
- a convex set of policy representations π,

such that these representations are embeddings in a vector space and the value of policy π under reward r at initial state q is a (simple) *multilinear* function of r, π, and q.

Our Contributions: Successor Feature Sets

For any control problem, we can write down

Note:

Given the embeddings, there are efficient algorithms to read off

- an optimal policy under a given reward function
- a policy that imitates a given set of demonstrations

How do we embed states?

(using Predictive states)

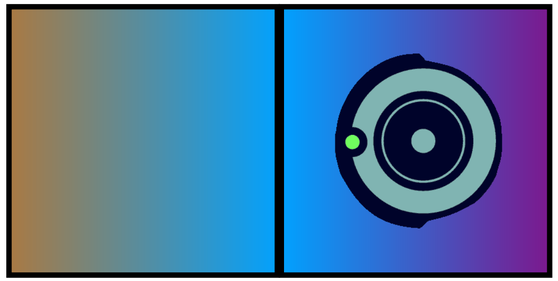
Changes the interpretation of the belief state:

Belief states no longer represent probability distribution over states, but instead satisfies two properties

Given starting state q_1, transition matrices T_{ao}, and a vector $u \in \mathbb{R}^k$

The two properties are:

1. $(\forall a, o, t) u^T T_{ao} q_t \geq 0$
2. $(\forall a, t) \sum_o u^T T_{ao} q_t = 1$



How do we embed states?

(using Predictive states)

Cl

tes,

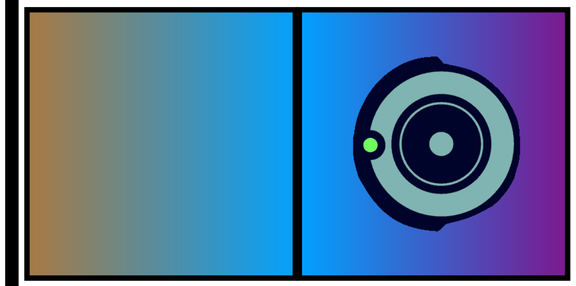
Represent state as a vector of predictions about the future

For example,

test #1: Probability ■ if you go left ?

test #2: Probability ■ if you go right ?

This is more general than belief states.

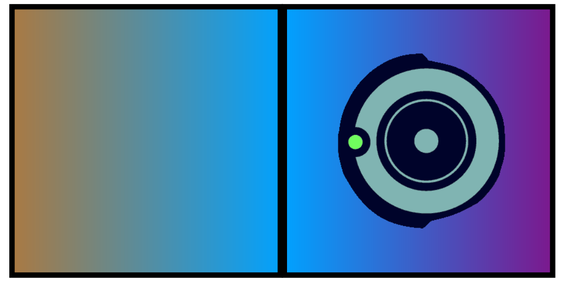


How do we embed rewards?

(using Successor Features)

Action-Value Function: **Falls apart when the reward function changes**

$$Q^{\pi}(q,a) = \mathbb{E}_{\pi}\left[\sum_{t=1}^{H} \gamma^{t-1} r(q_t, a_t) \,|\, \text{do } q_1 = q, a_1 = a\right]$$



How do we embed rewards?

(using Successor Features)

Suppose our reward function is defined by state-action features: $r(q,a) = r^T f(q,a)$ for some vector r and feature function f which is linear in q for each a

Let $f(q,a) = F_a q$ **Then** $Q^\pi(q) = \mathbb{E}_\pi \left[\sum_{t=1}^{H} \gamma^{t-1} r^T F_{a_t} q_t \,|\, \text{do } q_1 = q, a_1 = a \right]$

Pulling out r^T

$$Q^\pi(q) = r^T \phi^\pi(q) \quad \textbf{and} \quad \phi^\pi(q) = \mathbb{E}_\pi \left[\sum_{t=1}^{H} \gamma^{t-1} F_{a_t} q_t \mid \text{do } q_1 = q, a_1 = a \right]$$

How do we embed rewards?

Let $f(q, a) = F_a q$

(using Successor Features)

Successor feature vector - $\phi^{\pi}(q) = \mathbb{E}_{\pi}\left[\sum_{t=1}^{H} \gamma^{t-1} F_{a_t} q_t \mid \text{do } q_1 = q, a_1 = a\right]$

They allow us to compute state-values for π under any reward function.

$$\phi^{\pi}(q_t) = \mathbb{E}_{\pi}[f(q_t, a_t) + \gamma \phi^{\pi(a_t, o_t)}(q_{t+1})]$$

$$\phi^{\pi}(q_t) = \sum_{a,o} P(a \mid \pi) P(o \mid q_t, \text{do } a)[F_a q_t + \gamma \phi^{\pi(a,o)}(q_{t+1})]$$

Successor Features for Transfer in Reinforcement Learning André Barreto, Will Dabney, Rémi Munos, Jonathan J. Hunt, Tom Schaul, Hado van Hasselt, David Silver

Universal Successor Features for Transfer Reinforcement Learning Chen Ma, Dylan R. Ashley, Junfeng Wen, Yoshua Bengio

Successor Features Combine Elements of Model-Free and Model-based Reinforcement Learning Lucas Lehnert, Michael L. Littman

Combining Everything

(Successor Feature, PSR, POMDP Value Iteration)

We now have all we need to write value as a multilinear function of state, reward, and policy. As a reminder,

- Our state q represents a linear combination of some fixed initial test states.
- Our reward r represents a linear combination of some features of our current state and action.
- Our policy embedding Φ tells us how successor features depend on initial state.

Given these quantities our state-value is now

- $V = r^T \phi q \mid \phi \in \Phi$

How do we embed policies?

(using POMDP value iteration ideas)

The function ϕ^π is linear in q, meaning there exists a matrix $A^\pi \in \mathbb{R}^{dxk}$ such that $\phi^\pi(q) = A^\pi q$

$$\phi^\pi(q_t) = \sum_{a,o} P(a \mid \pi) P(o \mid q_t, \text{do } a)[F_a q_t + \gamma \phi^{\pi(a,o)}(q_{t+1})]$$

$$\phi^\pi(q_t) = \sum_{a} P(a \mid \pi) \left[F_a q_t + \gamma \sum_{o} A^{\pi(a,o)} T_{ao} q_t \right]$$

$$A^\pi = \sum_{a} P(a \mid \pi) \left[F_a + \gamma \sum_{o} A^{\pi(a,o)} T_{ao} \right]$$

Successor feature matrix - $A^\pi = F_a + \gamma \sum_{o} A^{\pi(o)} T_{ao}$

How do we embed policies?

$$A^{\pi} = F_a + \gamma \sum_o A^{\pi(o)} T_{ao}$$

(using POMDP value iteration ideas)

[Main new contribution]: Successor Feature Sets

$$\Phi^{(H)} = \{A^{\pi} \mid \pi \text{ a policy with horizon } H\}$$

Set satisfies Bellman Equations

$$\Phi_a^{(H)} = F_a + \gamma \sum_o \Phi^{(H-1)} T_{ao}$$

$$\Phi^{(H)} = \text{conv} \bigcup_a \Phi_a^{(H)}$$

Implementation Detail: Let $m_i \in \mathbb{R}^{d \times k}$ be a fix a set of directions then

$$\Phi^H \approx \arg\max \langle m_i, \phi \rangle \text{ for } \phi \in \bigcup_{a'} \left[F_{a'} + \gamma \sum_{o'} \Phi_{a'o'} \right] T_{ao}$$

Imitation Learning by feature Matching

(optimal planning)

- Given demonstration from a behavior policy

$$s_0, a_0, a_1, s_1, s_2, a_2, \ldots$$

- Compute expected discounted features

$$f(s_0, a_0) + \gamma^1 f(s_1, a_1) + \gamma^2 f(s_2, a_2) + \ldots$$

- We want to find a policy in our successor feature set that matches the behavior policy

$$\mathbb{E}_\pi \left[f(s_0, a_0) + \gamma^1 f(s_1, a_1) + \gamma^2 f(s_2, a_2) + \ldots \right] = \phi^d$$

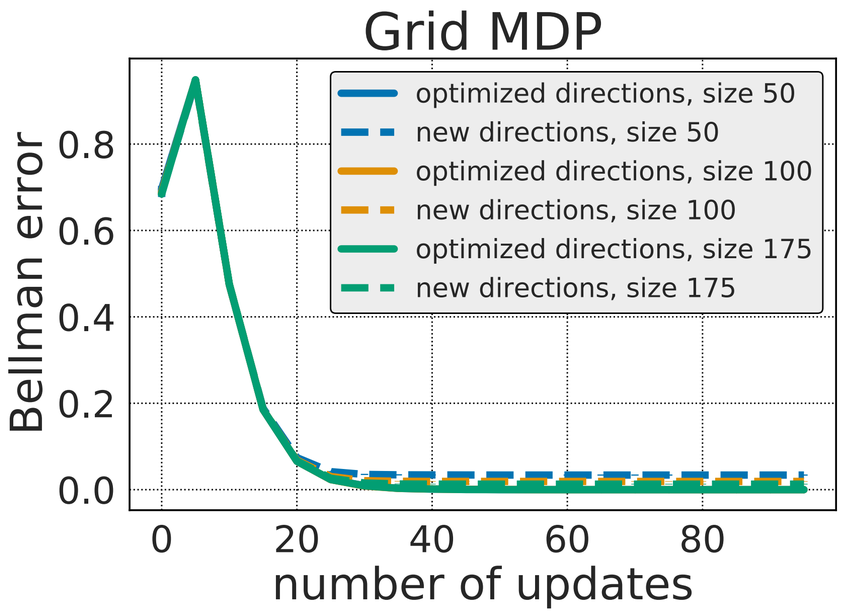
Experiment Details:

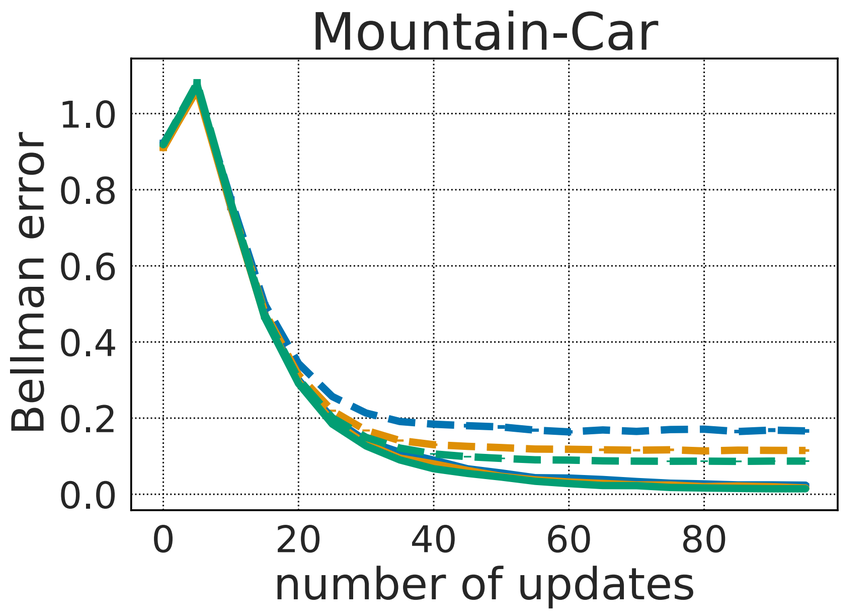
(testing our planning)

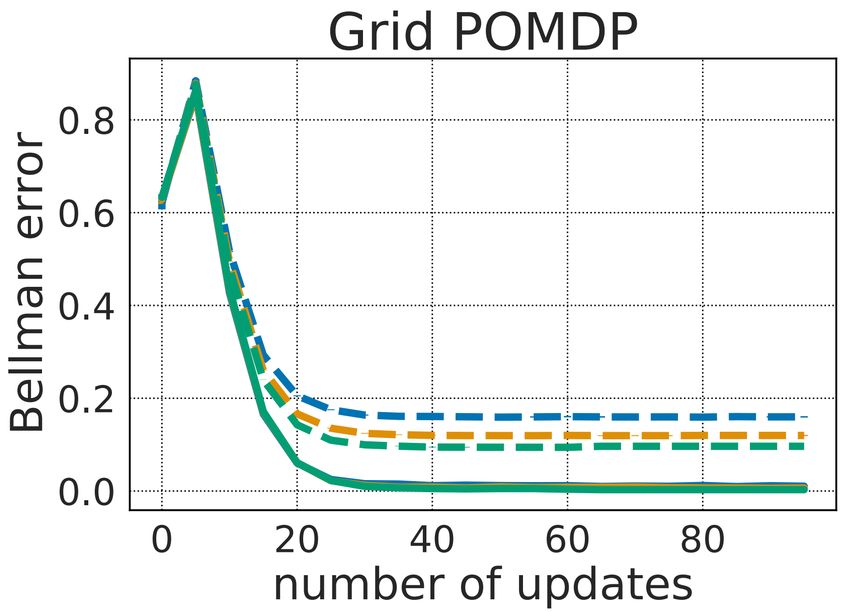
We looked at scaling behavior in three simple environments:

- **MDP:**
  - deterministic 18 × 18 gridworld
- **Mountain-Car:**
  - finite-element discretization of the standard mountain-car benchmark
- **POMDP:**
  - partially observable 18 x 18 gridworld (where transition dynamics and state observations are both slightly noisy)

Experiment Results:







Conclusion:

We showed for any control problem, we can write down

- a convex set of possible state representations q,
- a convex set of reward function representations r, and
- a convex set of policy representations π,

such that these representations are embeddings in a vector space

We evaluated out embedding on three task

Thank you