Sample-efficient Learning in Many-goals Settings

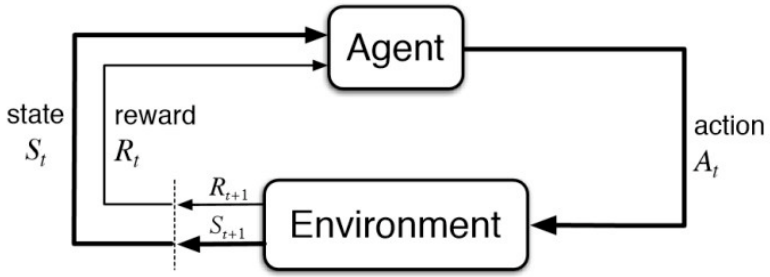
PRESENTOR: BYUNGSOO JEON

LLL READING GROUP @ CMU

Reminder: RL

Reinforcement Learning

- Given the environment, how do we learn the sequence of actions (policy) to maximize the cumulative reward?



Value Function, $V_\pi(s) = \mathbb{E}_\pi \left[R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+3} + \ldots \mid S_t = s \right]$

- Expected total reward for an agent starting from state s

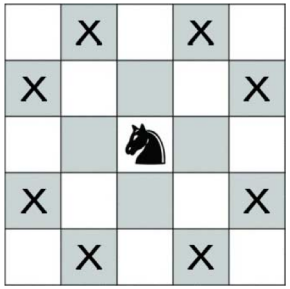
Reminder: Task vs. Goal

Task

- Def: A piece of work to be done
- e.g. Beating an opponent in the chess (or Starcraft)

Goal

- Def: The end toward which effort is directed
- e.g. Move the knight to one of good positions



Hindsight Experience Replay

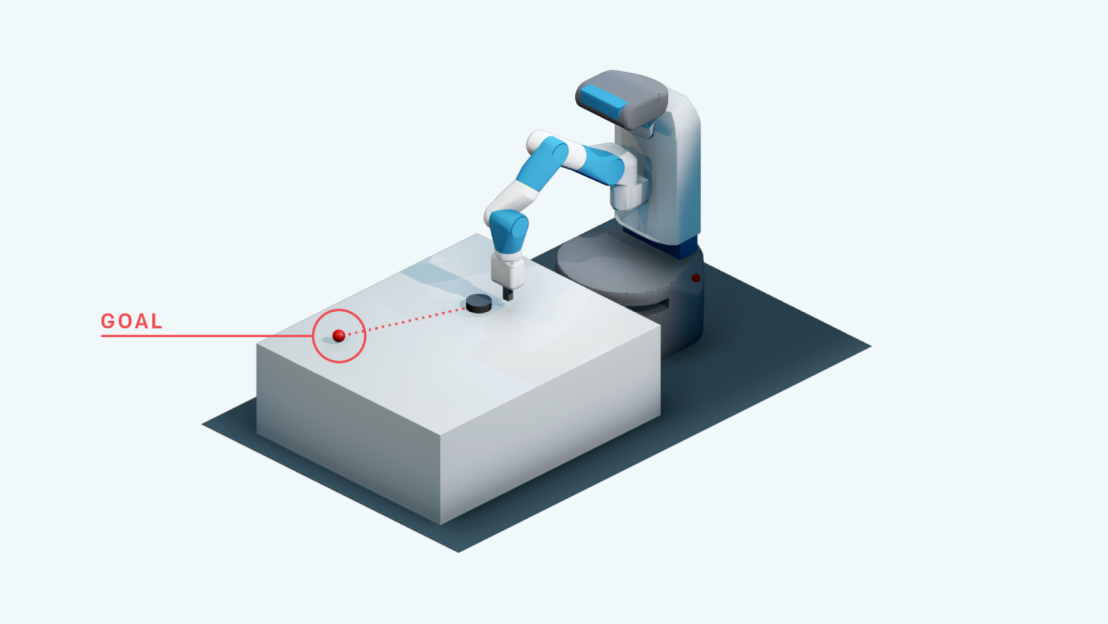
PRESENTOR: BYUNGSOO JEON

LLL READING GROUP @ CMU

Motivation

Dealing with sparse rewards

- How can we evade the need for complicated reward engineering?



Problem Statement

In the many-goals setting, how can we achieve sample-efficient learning even if the reward signal is binary and sparse?

Assumptions

- We use an off-policy RL algorithm
- The goals influence the agent's actions, but not the environment dynamics

Proposed Method

Hindsight **E**xperience **R**eplay (a.k.a. **HER**)

Key idea

- Replay each episode with a different goal than the one the agent was trying to achieve, e.g, one of the goals already achieved in the episode (Hindsight)

Hypothesis

- Replaying with a different goal plays the role of implicit curriculum in a way to facilitate learning

Proposed Method

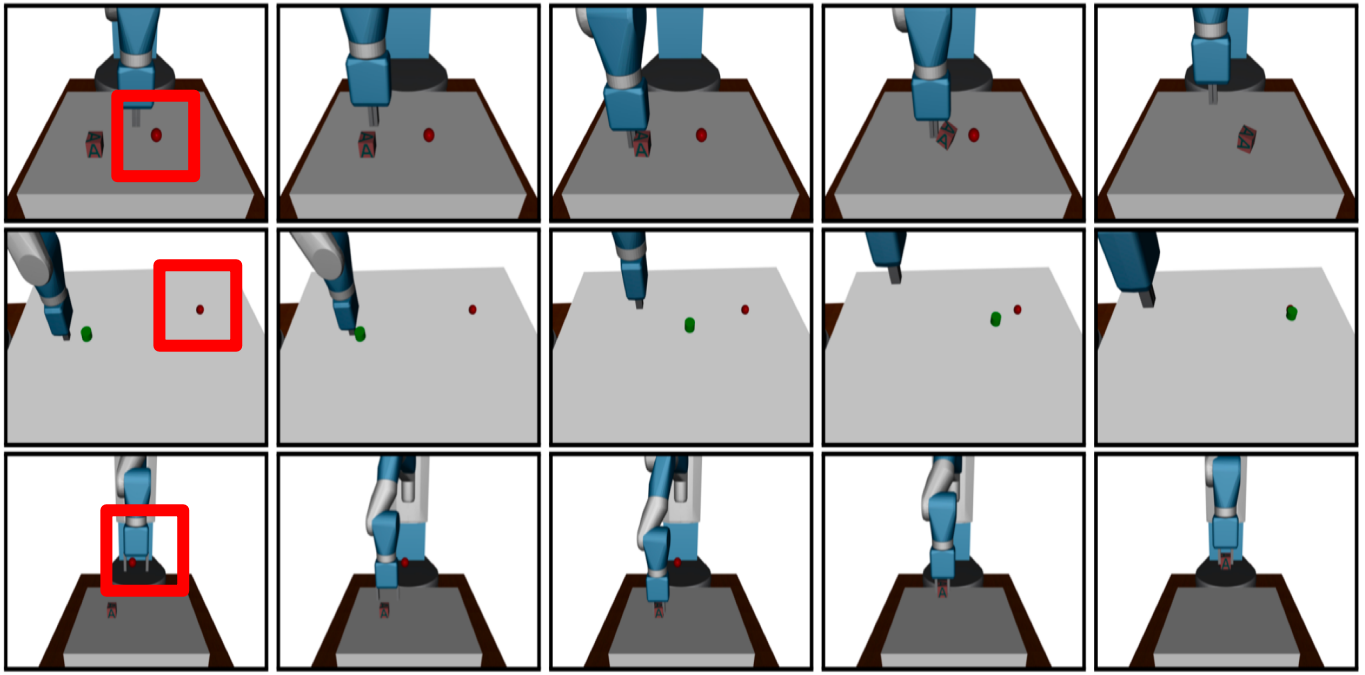
Algorithm 1 Hindsight Experience Replay (HER)

Given:

- an off-policy RL algorithm $\mathbb{A}$, ▷ e.g. DQN, DDPG, NAF, SDQN
- a strategy $\mathbb{S}$ for sampling goals for replay, ▷ e.g. $\mathbb{S}(s_0, \ldots, s_T) = m(s_T)$
- a reward function $r : \mathcal{S} \times \mathcal{A} \times \mathcal{G} \rightarrow \mathbb{R}$. ▷ e.g. $r(s, a, g) = -[f_g(s) = 0]$

Initialize $\mathbb{A}$ ▷ e.g. initialize neural networks
Initialize replay buffer R
for episode = 1, M **do**
  Sample a goal g and an initial state s_0.
  for $t = 0, T-1$ **do**
    Sample an action a_t using the behavioral policy from $\mathbb{A}$:
      $a_t \leftarrow \pi_b(s_t||g)$ ▷ $||$ denotes concatenation
    Execute the action a_t and observe a new state s_{t+1}
  end for
  for $t = 0, T-1$ **do**
    $r_t := r(s_t, a_t, g)$
    Store the transition $(s_t||g,\ a_t,\ r_t,\ s_{t+1}||g)$ in R ▷ standard experience replay
    Sample a set of additional goals for replay $G := \mathbb{S}$(**current episode**)
    for $g' \in G$ **do**
      $r' := r(s_t, a_t, g')$
      Store the transition $(s_t||g',\ a_t,\ r',\ s_{t+1}||g')$ in R ▷ HER
    end for
  end for
  for $t = 1, N$ **do**
    Sample a minibatch B from the replay buffer R
    Perform one step of optimization using $\mathbb{A}$ and minibatch B
  end for
end for

Experimental Setting



Pushing

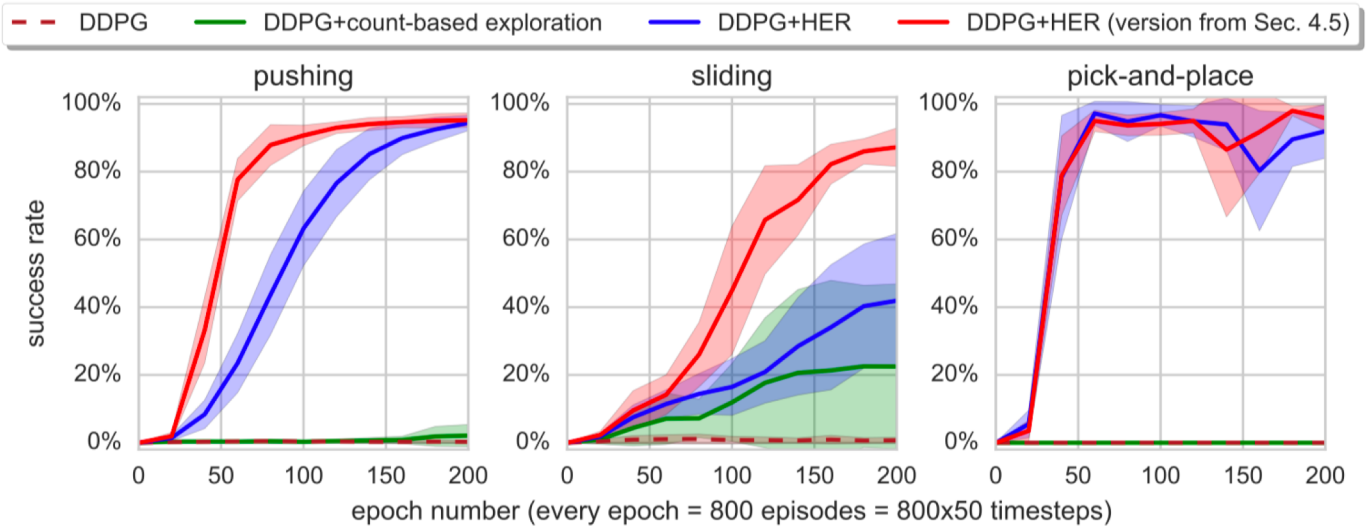
Sliding

Pick-and-place

State	Goal	Reward
Physical info	Desired position of the object	$f_g(s) = [\lvert g - s_{obj} \rvert \leq \epsilon]$

Result

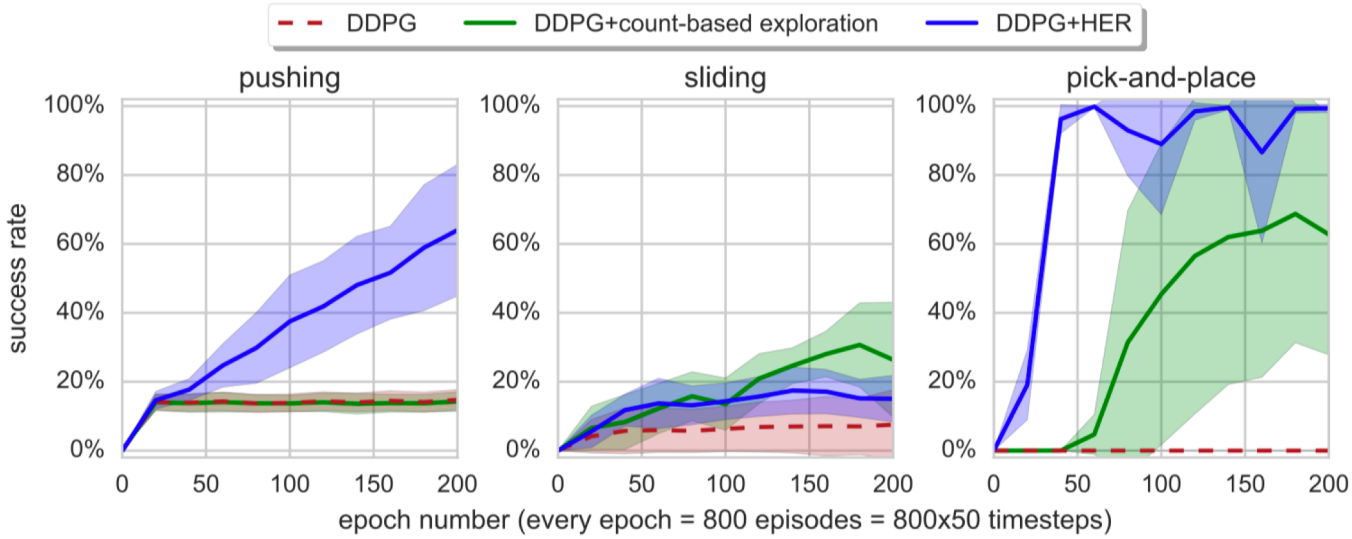
Does HER improve performance?



- Yes. DDPG+HER improves over baselines while baselines fail to solve (Pushing, Pick-and-Place) or makes slower progress (Sliding)

Result

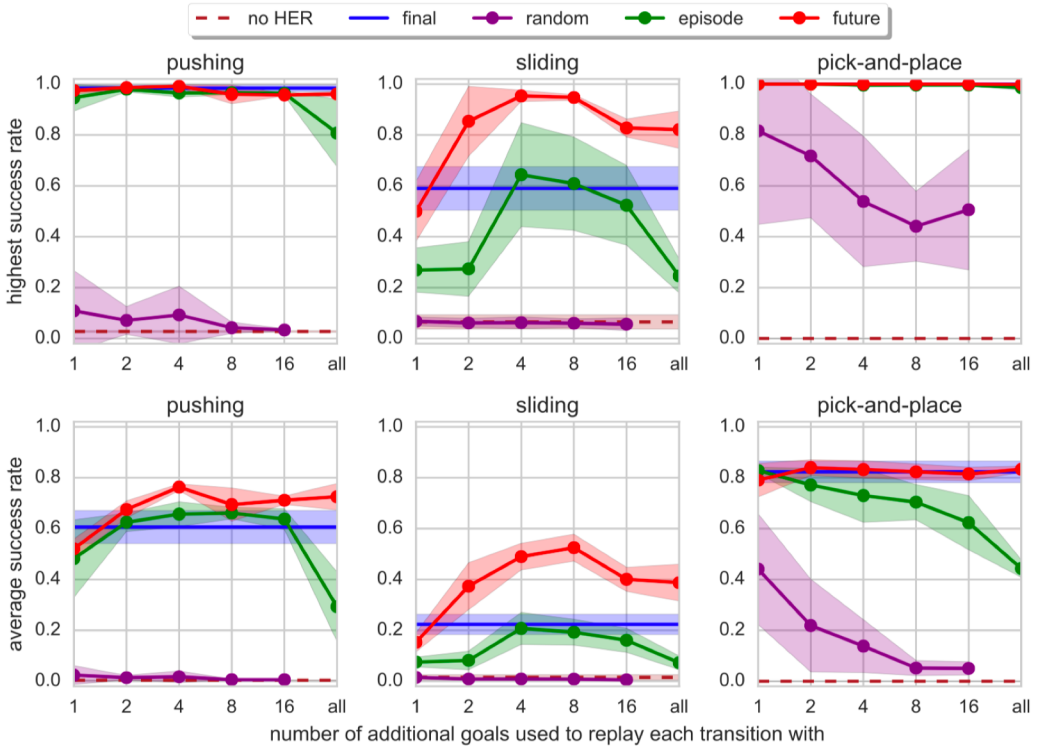
Does HER improve performance **even if there is a single goal we care about?**



- Yes. DDPG+HER is better than DDPG while DDPG fails to solve (Pushing, Pick-and-Place) or makes slower progress (Sliding)

Result

How many goals should we replay each trajectory with and how to choose them ?



Result

Trained policy in the simulator solves the pick-and-place task well on the physical robot without any finetuning

Conclusion

HER enables sample-efficient learning using only sparse and binary rewards

We can combine HER with arbitrary off-policy RL algorithms

As a result, HER successfully learns complicated behavior three challenging tasks that vanilla RL algorithm fails to solve

- Note that it is the first work to solve these tasks only with binary and sparse reward

Universal Value Funnction Approximators

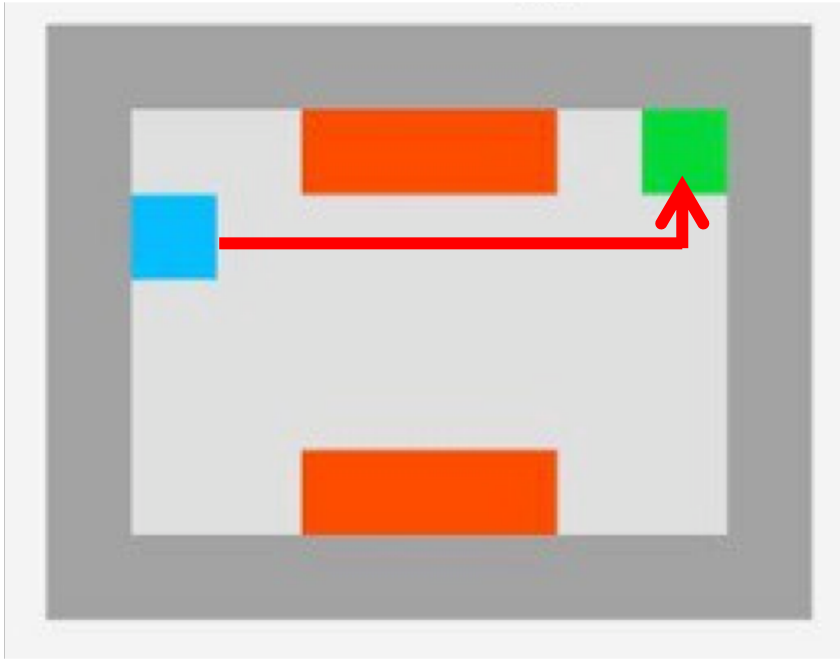
PRESENTOR: BYUNGSOO JEON

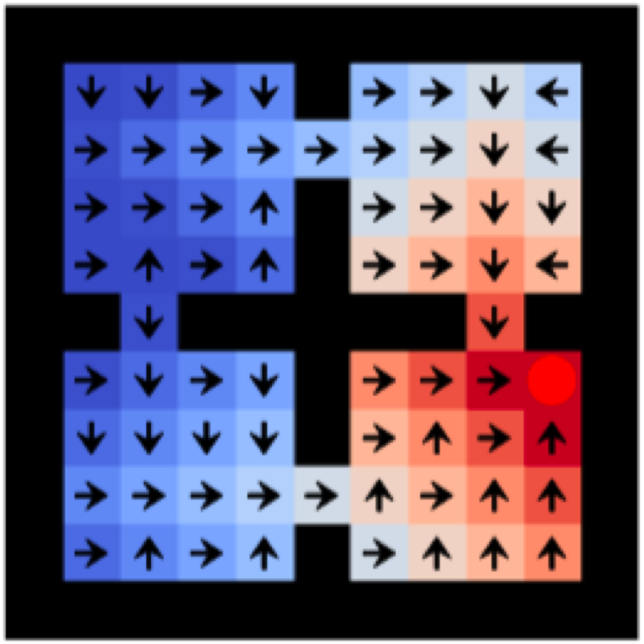
LLL READING GROUP @ CMU

Motivation

Dealing with sparse rewards

- How can we exploit the structure in the goal space?





Problem Statement

In the many-goals setting, how can we achieve sample-efficient learning by **exploiting the structure in the goal space**?

Assumptions

- The goal space contains enough structural information to help generalization to the value of similar, unseen goals
  - Similarity encoded a priori in the goal representation
  - The structure in the induced value functions

Proposed Method

Universal **V**alue **F**unction **A**pproximators (UVFA)

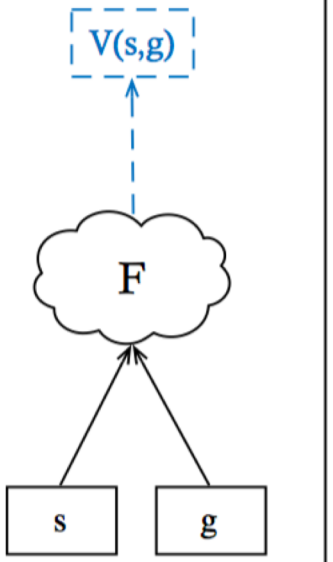
Key idea

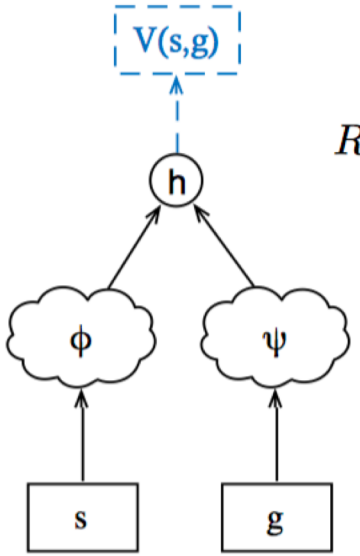
- Extend the idea of value function approximation $V_\theta(s)$ to both state and goal, $V_\theta(s, \boldsymbol{g})$

Hypothesis

- UVFA is capable of exploiting the structural information in the goal space
- UVFA allows the generalization to unseen states and goals

Proposed Method



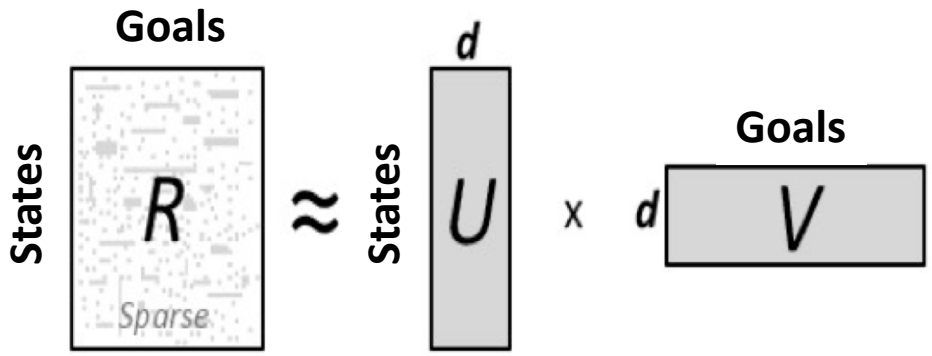


$$R_g(s,a,s') = \begin{cases} 1, & s' = g \text{ and } \gamma_{ext}(s) \neq 0 \\ 0, & \text{otherwise} \end{cases}$$

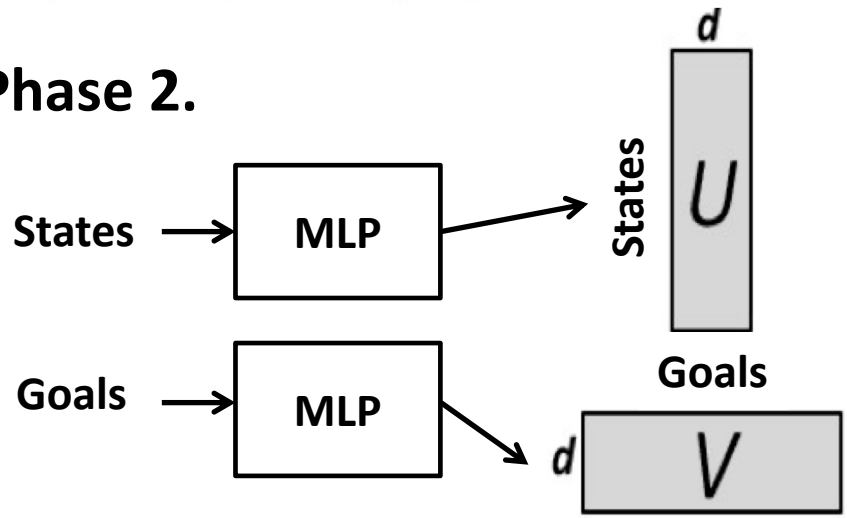
Goal: Represent a large set of optimal value functions by a single, unified function approximator that generalize over both states and goals

Proposed Method

Phase 1.



Phase 2.



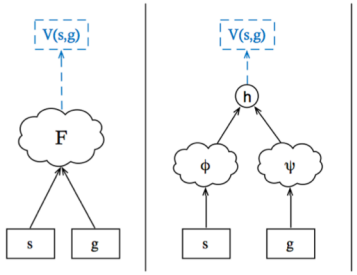
Algorithm 1 UVFA learning from Horde targets

Input: rank n, training goals $\mathcal{G}_T$, budgets b_1, b_2, b_3
Initialise transition history $\mathcal{H}$
for $t = 1$ **to** b_1 **do**
  $\mathcal{H} \leftarrow \mathcal{H} \cup (s_t, a_t, \gamma_{ext}, s_{t+1})$
end for
for $i = 1$ **to** b_2 **do**
  Pick a random transition t from $\mathcal{H}$
  Pick a random goal g from $\mathcal{G}_T$
  Update Q_g given a transition t
end for
Initialise data matrix $\mathbf{M}$
for $(s_t, a_t, \gamma_{ext}, s_{t+1})$ **in** $\mathcal{H}$ **do**
  for g **in** $\mathcal{G}_T$ **do**
    $\mathbf{M}_{t,g} \leftarrow Q_g(s_t, a_t)$
  end for
end for
Compute rank-n factorisation $\mathbf{M} \approx \boldsymbol{\phi}^\top \boldsymbol{\psi}$
Initialise embedding networks ϕ and ψ
for $i = 1$ **to** b_3 **do**
  Pick a random transition t from $\mathcal{H}$
  Do regression update of $\phi(s_t, a_t)$ toward $\hat{\boldsymbol{\phi}}_t$
  Pick a random goal g from $\mathcal{G}_T$
  Do regression update of $\psi(g)$ toward $\hat{\boldsymbol{\psi}}_g$
end for
return $Q(s, a, g) := h(\phi(s, a), \psi(g))$

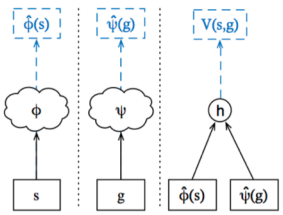
Result

Does **two-stream architecture** for UVFA outperform other baselines?

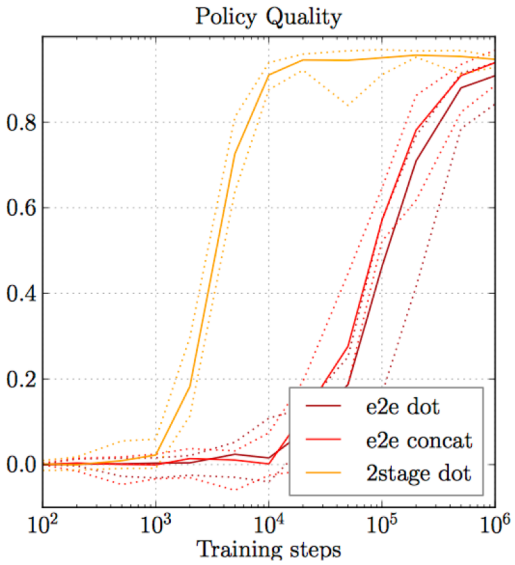
e2e concat

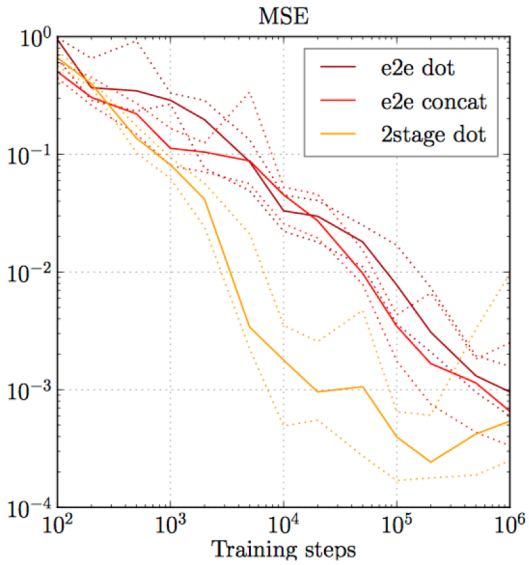


2stage dot



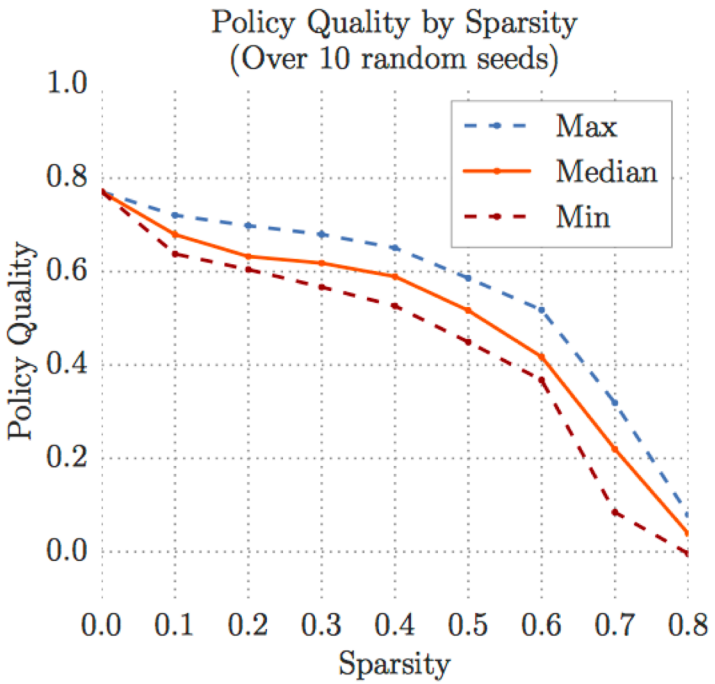
e2e dot

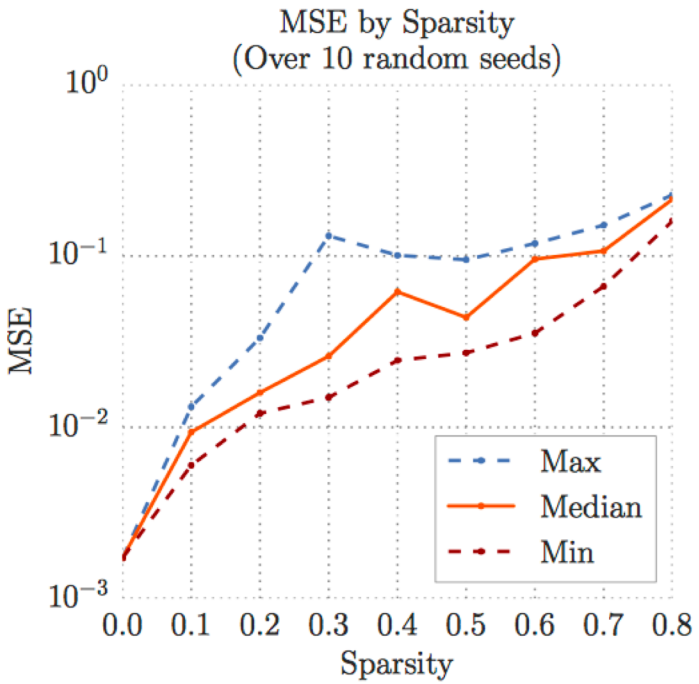




Result

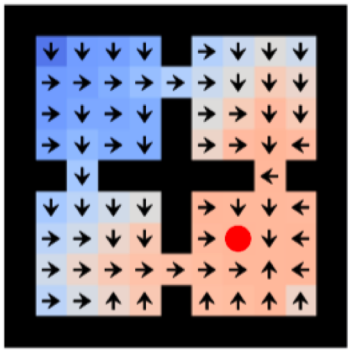
How robustly does UVFA **generalizes** to the **unseen states** and **goals** across different **sparsity** of the value table?

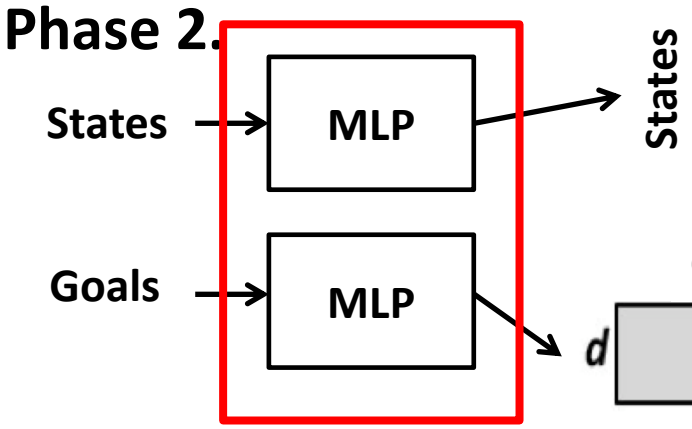


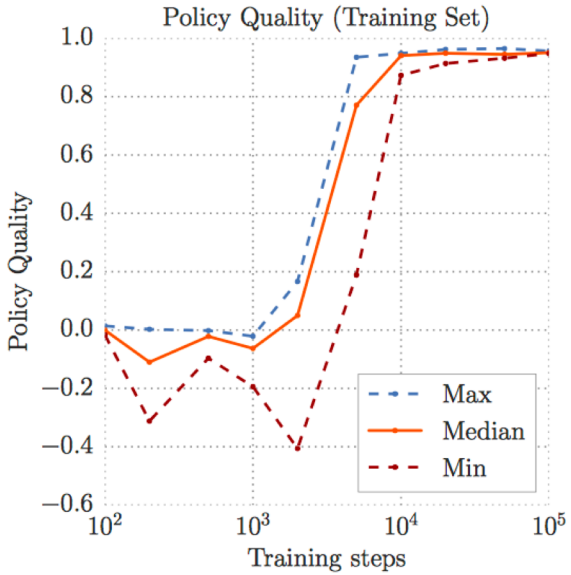


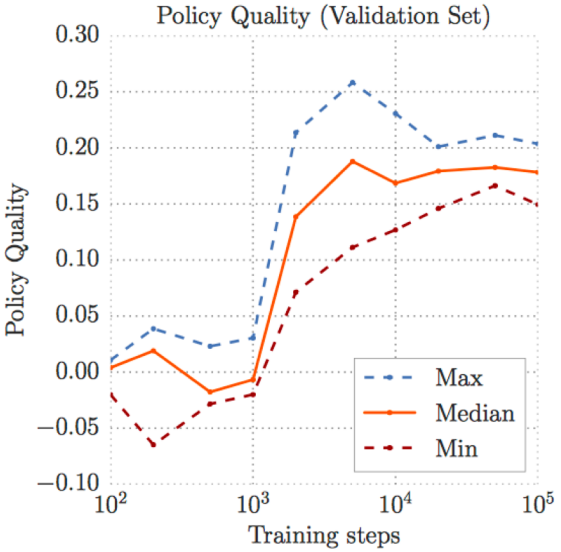
Result

How well does UVFA **interpolate** the values for **unseen goals**?



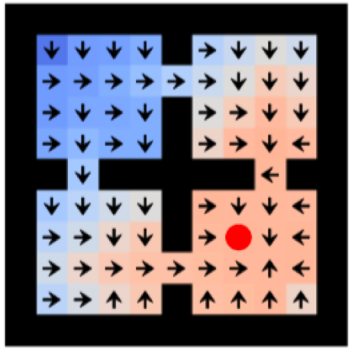


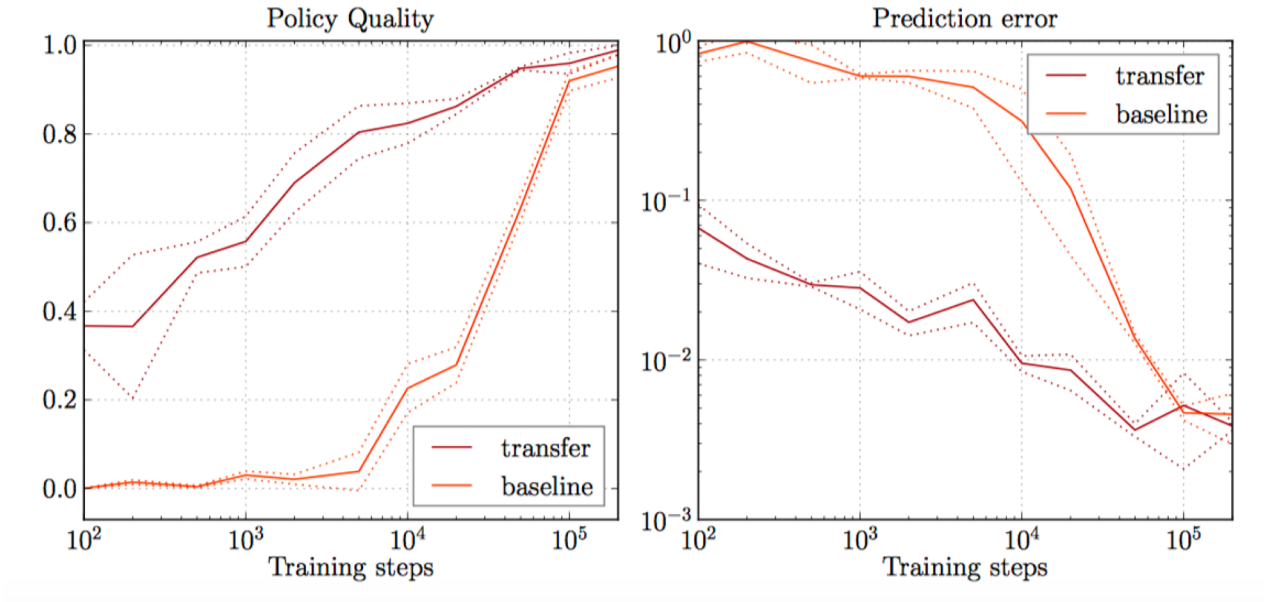




Result

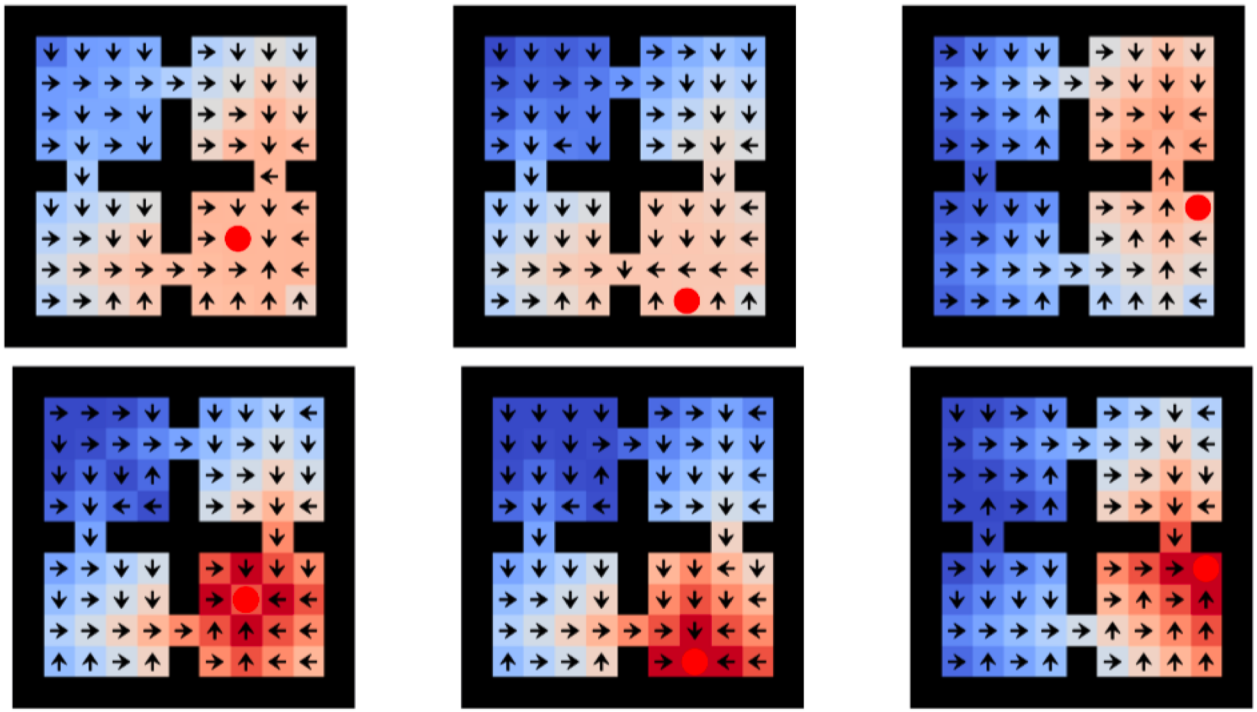
Can UVFA be used for transfer learning to new tasks with the same dynamics, but different goals?





Result

How well does UVFA **extrapolates** the values for **unseen goal**? (e.g. trained only on the goals except bottom-right room)



Conclusion

UVFA can successfully generalize to previously unseen goals and states

- For both interpolation and extrapolation
- For both supervised and reinforcement learning settings

UVFAs can be used for **transfer learning** to new tasks with the same dynamics, but different goals

Generalized value functions can be used as **features** to represent state (predictive representation)

References

[MFA17] Marcin Andrychowicz et al., “Hindsight Experience Replay”, NIPS 2017

[TDK15] Tom Schaul et al., “Universal Value Function Approximators”, ICML 15

Question or Discussion