Strategic Exploration via State Abstraction from Rich Observations

John Langford

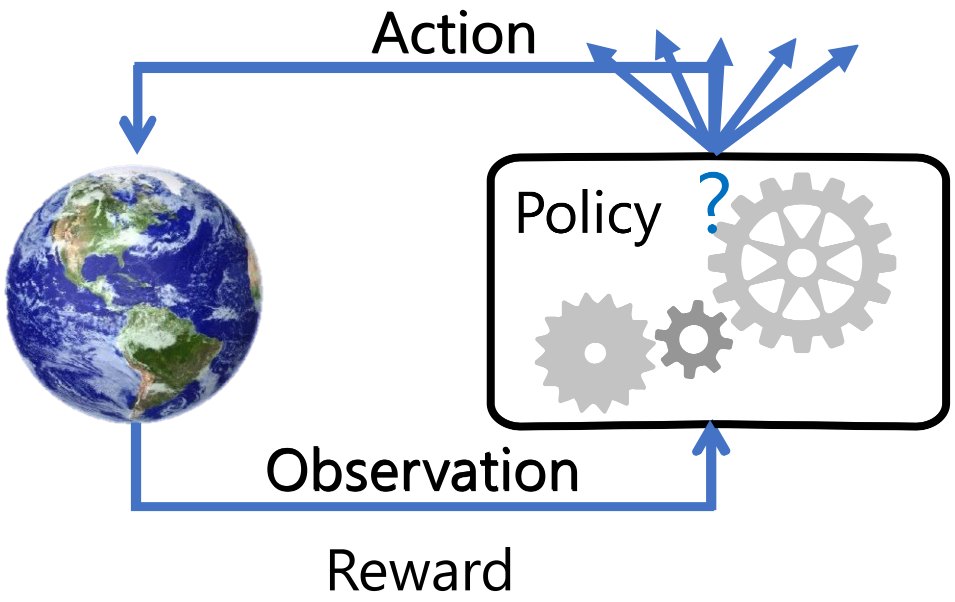
MSR-New York City

Forthcoming work with

Dipendra Misra Mikael Henaff Akshay Krishnamurthy

Fresher than Arxiv! https://tinyurl.com/msr-homer

Reminder: Reinforcement Learning



Goal: Find a policy maximizing the sum of rewards

What's hard?

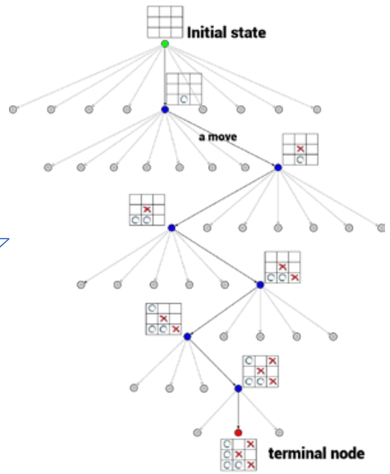
Policy Improvement

Generalization

??

Contextual Bandits

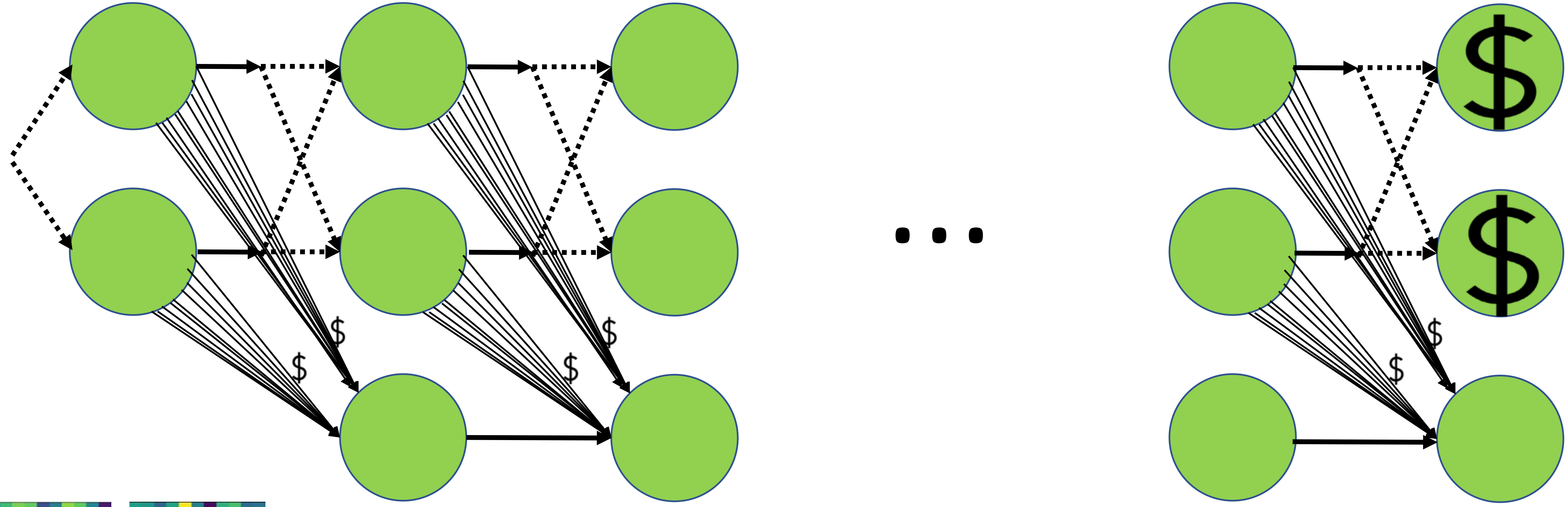
MDP Learning

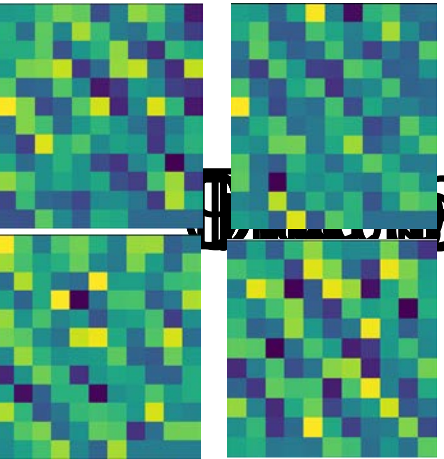


Credit

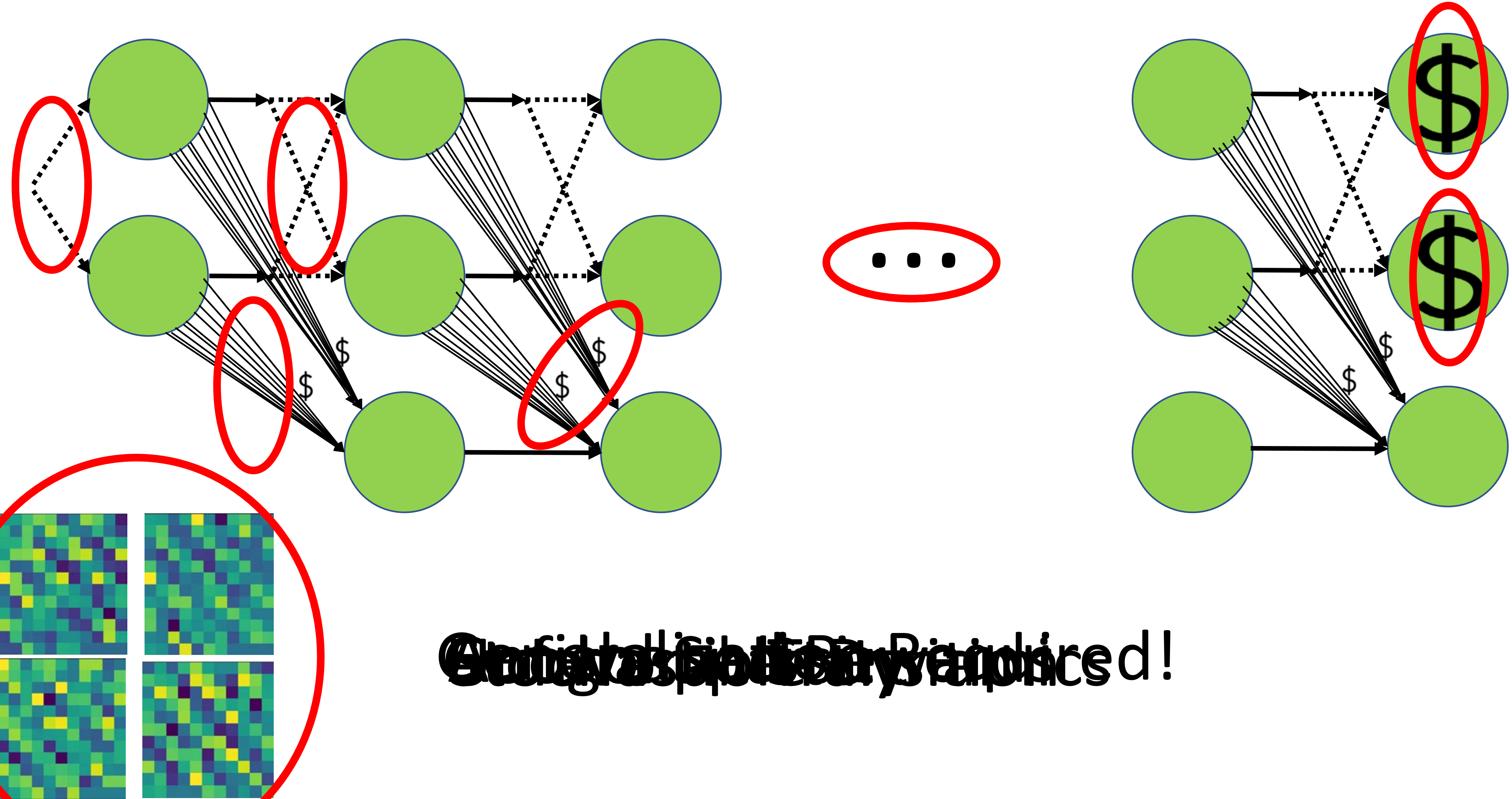
Exploration

Hard Reinforcement Learning problem





Why is this hard?



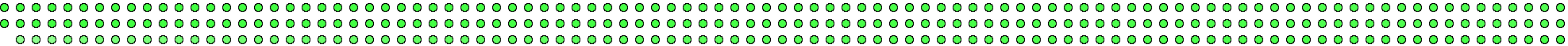
State Visitation of Common RL algorithms

Advantage Actor Critic (A2C)

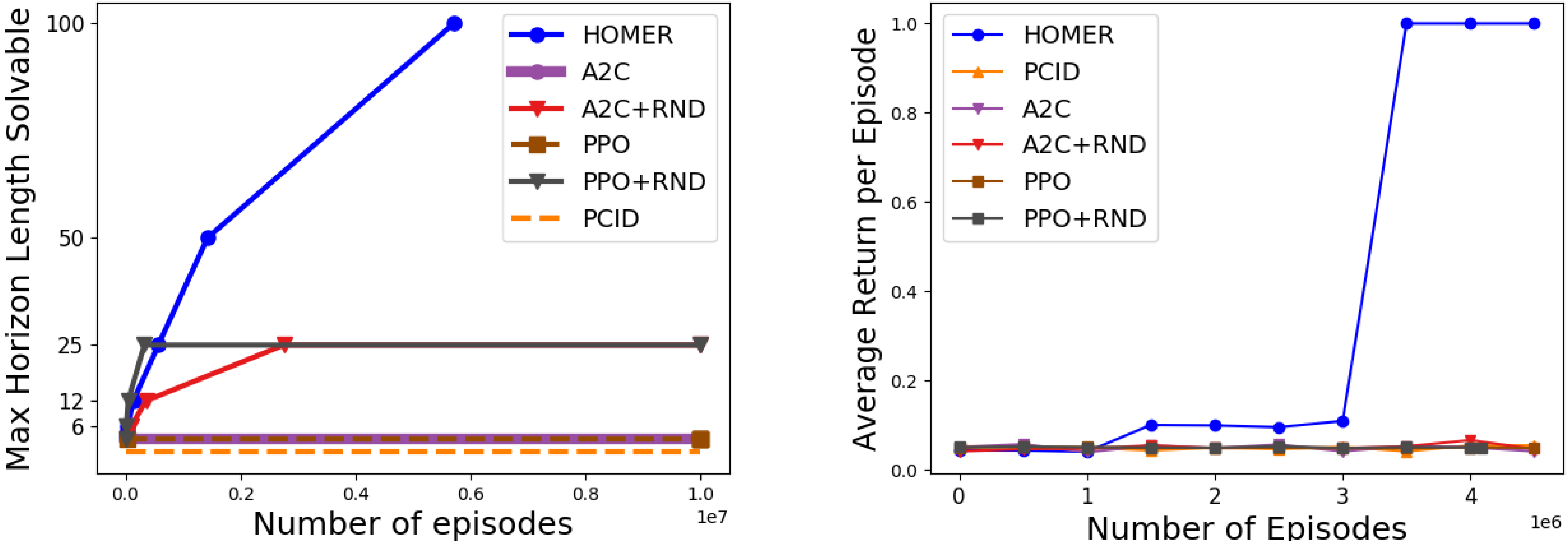
Proximal Policy Optimization (PPO)

Random Network Distillation (RND)

Homer (New!)



Performance



How do you formulate the problem?

Repeatedly

For h = 1 to H

See observation $x \in R^n$

Generated by some latent state s.

Never repeats!

Choose action $a \in \{1, \dots, K\}$

Causes stochastic transition to latent state s'.

See reward $r \in [0,1]$

Generated by x, a, and next observation x'.

This could be per action or per episode.

Goal: Compete with some policy class $\Pi = \{\pi : x \to a\}$

Two assumptions

Block MDP: For all observations x there is a unique s which can generate it.

Oracle Learning: Supervised learning problems can be solved well with sufficient data.

Theorem: Homer solves all Block MDP problems with $\text{poly}(|S|, |A|, H)$ samples and time if Oracle Learning works.

Independent of $|X|$!

Key Concept: Kinematic State

Kinematic State = observations with same causal dynamics.

Backward kinematic state:

$x_1', x_2' \in s$ if for all $\mathrm{u} \in \Delta(x, a)$,

Forward kinematic state:

$x_1, x_2 \in s$ if for all x', a:

$$T(x'|x_1, a) = T(x'|x_2, a)$$

Kinematic state = Forward+Backward

Key Concept: Homing Policy

Homing Policy = policy finding something with highest probability.

$$\text{For all } x\text{: } \pi_x = \operatorname*{argmax}_{\pi} P_\pi(x)$$

$$\text{For all } s\text{: } \pi_s = \operatorname*{argmax}_{\pi} P_\pi(s)$$

Kinematic state $s \Rightarrow$ every $x \in s$ homed by same policy.

Homer

For each h=2 to H

Many times

Sample π ~ Uniform (policy cover Π_{h-1})

(x, a, x') ~ h-1 steps with π then act uniform random

50% -> keep $(x, a, x', 1)$ else keep $(x, a, \mathrm{Uniform}(\{x'\}), 0)$

Learn to predict whether x' corrupted.

$$(\mathrm{p}, \phi, \phi') = \mathrm{argmin}_{p,\phi,\phi'} \hat{E}_{(x,a,x',y)} \left(p\left(\phi(x), a, \phi'(x')\right) - y\right)^2$$

For each value of bottleneck $\mathrm{s} = \phi'(x')$

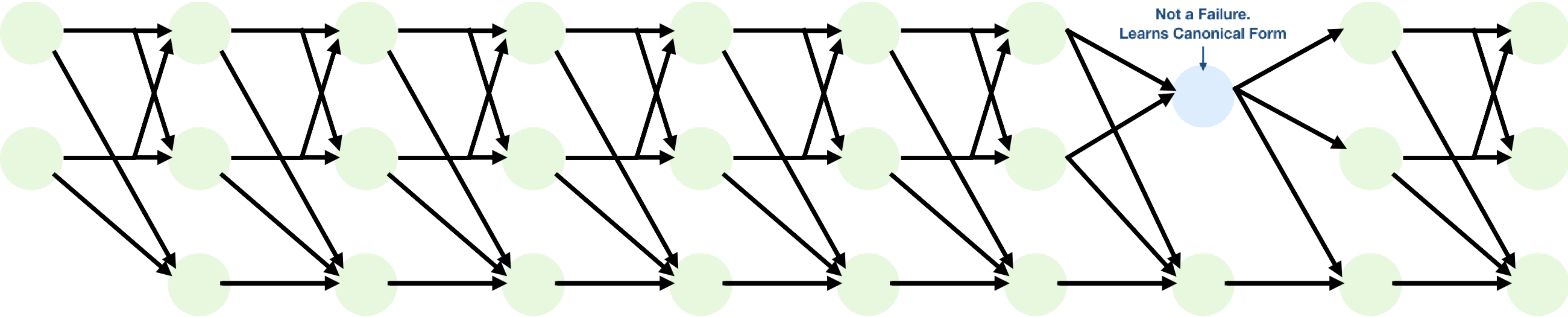
Define Reward $R_s(x', a) = I(\phi'(x') = s)$

Learn homing policy $\pi_s = \mathrm{Find\_Policy}(\{\Pi_i\}, R_s)$

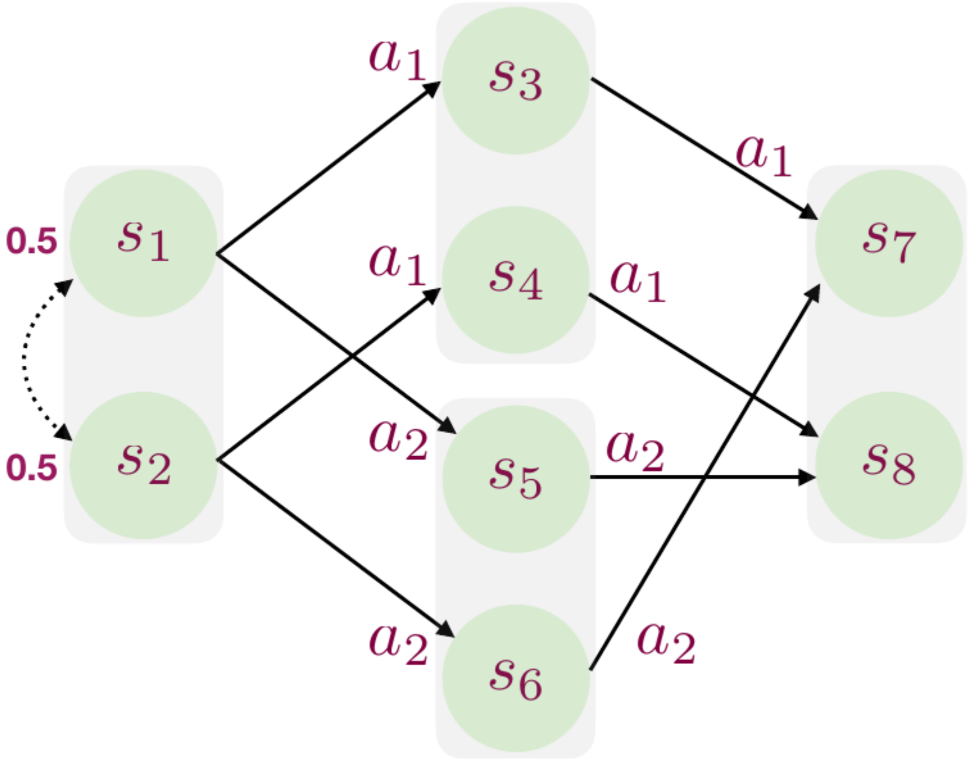
Form policy cover $\Pi_h = \{\pi_s\}$

Return Policy cover $\{\Pi_i\}$

We can extract the underlying state space!



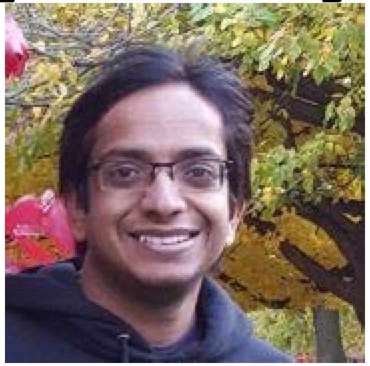
A good example to think about



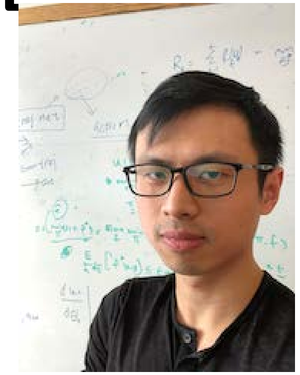
1. ~~Predict action given x, x'~~
2. ~~Predict action + previous state given x'~~
3. ~~Construct homing policies incrementally~~

Past and Future Work

[KAL16] [JKALS16] [DJAKLS18] [SJKAL19] [DKJADL19]







Active Research area!

How do we make the algorithm incremental?

How do we handle continuous state/action?

How do we handle combinatorial state?

Yes, we are hiring!

Many people, locations, roles:
http://aka.ms/rl_hiring