RL^2 FAST REINFORCEMENT LEARNING VIA SLOW REINFORCEMENT LEARNING

Presenters : Kshama Nitin Shah , Kemmannu Vineet Rao

Duan, Yan, John Schulman, Xi Chen, Peter L. Bartlett, Ilya Sutskever, and Pieter Abbeel. 'RL2: Fast Reinforcement Learning via Slow Reinforcement Learning'. *ArXiv [Cs.AI]*, 2016. arXiv. http://arxiv.org/abs/1611.02779

Motivation

- Animals learn new tasks in few trials

- Usage of prior knowledge to learn new tasks quickly

Duan, Yan, John Schulman, Xi Chen, Peter L. Bartlett, Ilya Sutskever, and Pieter Abbeel. ‘RL2: Fast Reinforcement Learning via Slow Reinforcement Learning’. *ArXiv [Cs.AI]*, 2016. arXiv. http://arxiv.org/abs/1611.02779
https://www.piedmont.org/living-better/the-mind-body-benefits-of-learning-a-new-skill

Motivation
RL is slow

Bellemare, Marc G., Sriram Srinivasan, Georg Ostrovski, Tom Schaul, David Saxton, and Rémi Munos. ‘Unifying Count-Based Exploration and Intrinsic Motivation’. *CoRR* abs/1606.01868 (2016)

Introduction

- Build an agent that can solve new and difficult tasks quickly
- Addresses the problem of sample inefficiency in RL
- Meta learning algorithm, objective : Learning Process
- Agent - designed as a recurrent neural network
- Generalizes to new tasks quickly

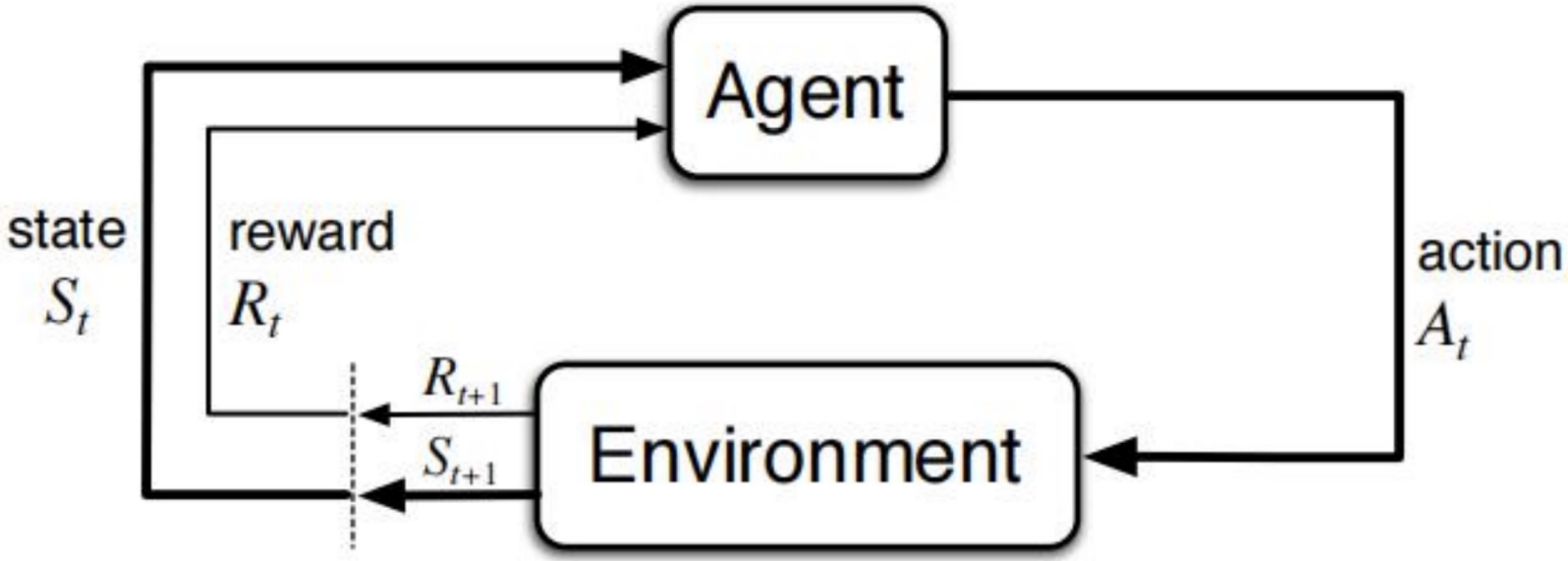
Related Work

PILCO

- Bayesian RL : Framework for incorporating prior knowledge into the learning process
- Limitation: Bayesian update is intractable in all but the simplest cases
- Thus, PILCO incorporates a mixture of Bayesian and domain specific ideas - reduces sample complexity
- Learns tasks in a few hours or few minutes even
- Drawback: Makes assumptions about the environment and computationally intractable in higher dimensions

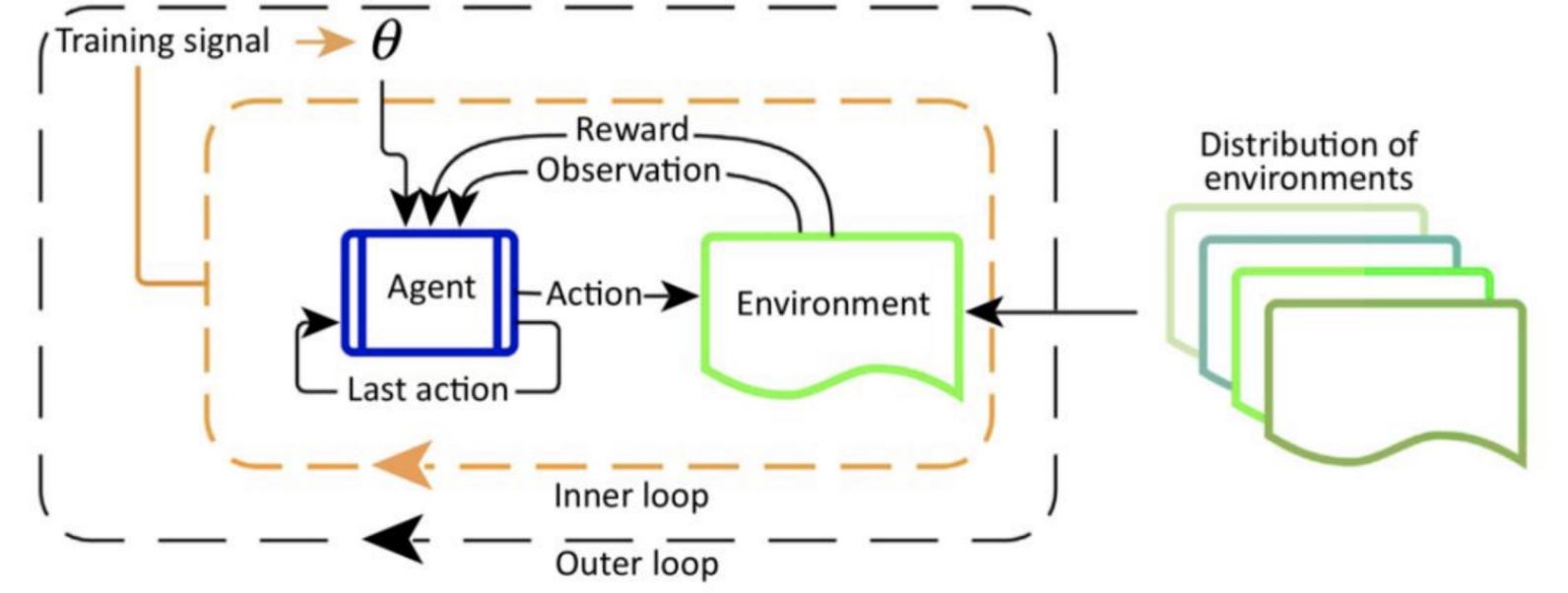
Deisenroth, M. P., and C. E. Rasmussen. 'PILCO: A Model-Based and Data-Efficient Approach to Policy Search'. In *Proceedings of the 28th International Conference on Machine Learning, ICML 2011*, edited by L. Getoor And Scheffer, 465–72. Omnipress, 2011.

Reinforcement Learning



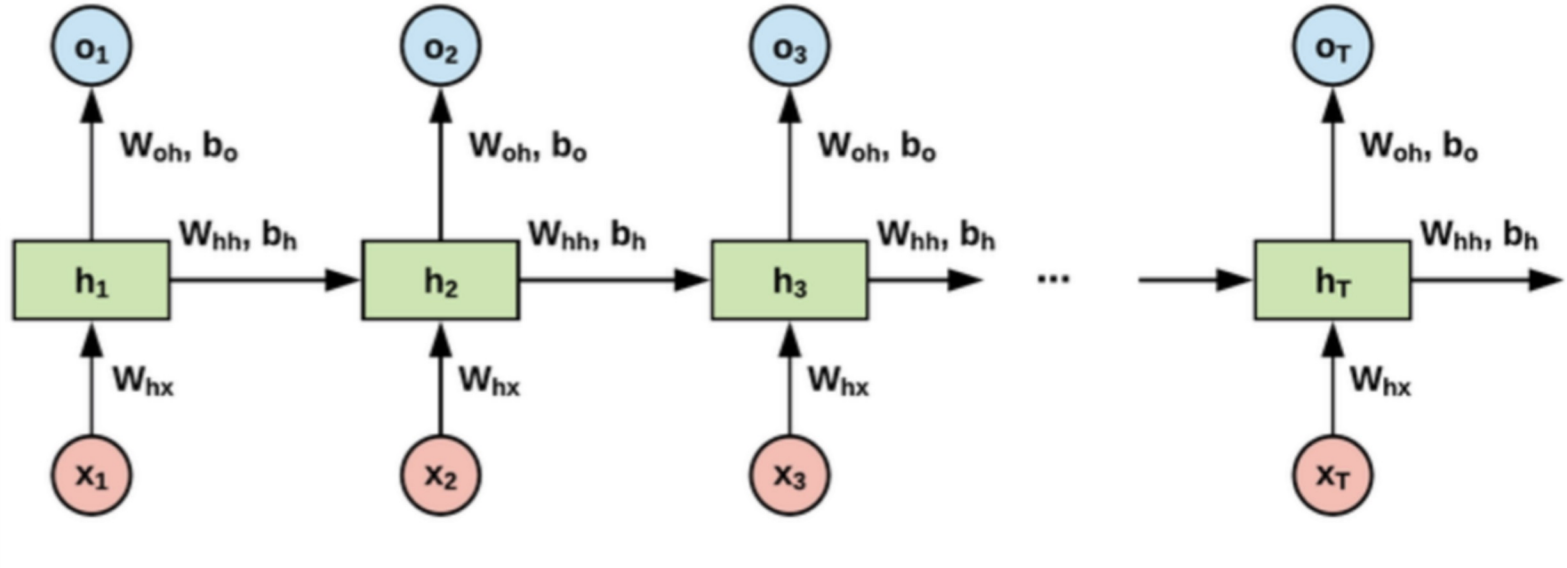
https://ml.cs.tsinghua.edu.cn/~chengyang/reading_meeting/Reading_Meeting_20220311.pdf

Meta Reinforcement Learning



https://ml.cs.tsinghua.edu.cn/~chengyang/reading_meeting/Reading_Meeting_20220311.pdf

RNN



https://disco.ethz.ch/courses/fs19/seminar/talks/meta_learning.pdf

Markov Decision Process (MDP)

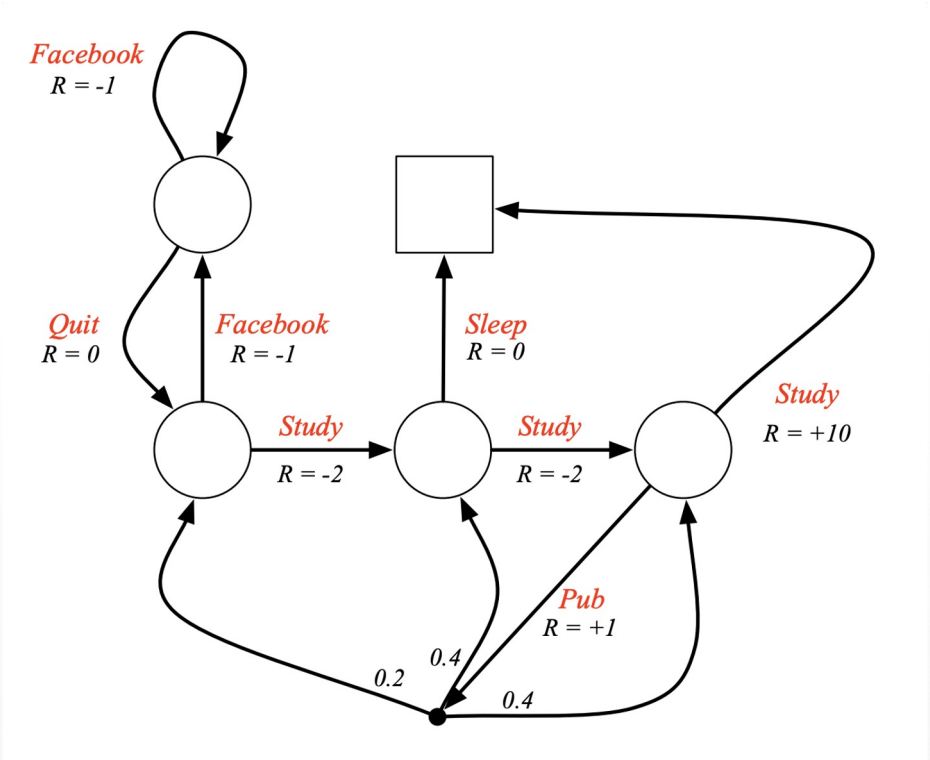
- Used to model decision-making problems where the outcome of a decision depends on the current state of the system and the chosen action.

Definition

A *Markov Decision Process* is a tuple $\langle \mathcal{S}, \mathcal{A}, \mathcal{P}, \mathcal{R}, \gamma \rangle$

- $\mathcal{S}$ is a finite set of states
- $\mathcal{A}$ is a finite set of actions
- $\mathcal{P}$ is a state transition probability matrix, $\mathcal{P}^{a}_{ss'} = \mathbb{P}\left[S_{t+1} = s' \mid S_t = s, A_t = a\right]$
- $\mathcal{R}$ is a reward function, $\mathcal{R}^{a}_{s} = \mathbb{E}\left[R_{t+1} \mid S_t = s, A_t = a\right]$
- γ is a discount factor $\gamma \in [0, 1]$.

Example: Student MDP



Partially Observable Markov Decision Process (POMDP)

- A Partially Observable Markov Decision Process is an MDP with hidden states. It is a hidden Markov model with actions.

Definition

A *POMDP* is a tuple $\langle \mathcal{S}, \mathcal{A}, \mathcal{O}, \mathcal{P}, \mathcal{R}, \mathcal{Z}, \gamma \rangle$

- $\mathcal{S}$ is a finite set of states
- $\mathcal{A}$ is a finite set of actions
- $\mathcal{O}$ is a finite set of observations
- $\mathcal{P}$ is a state transition probability matrix, $\mathcal{P}^{a}_{ss'} = \mathbb{P}\left[S_{t+1} = s' \mid S_t = s, A_t = a\right]$
- $\mathcal{R}$ is a reward function, $\mathcal{R}^{a}_{s} = \mathbb{E}\left[R_{t+1} \mid S_t = s, A_t = a\right]$
- $\mathcal{Z}$ is an observation function, $\mathcal{Z}^{a}_{s'o} = \mathbb{P}\left[O_{t+1} = o \mid S_{t+1} = s', A_t = a\right]$
- γ is a discount factor $\gamma \in [0, 1]$.

Policy Search Methods

- A policy Π is a distribution over actions given states

Definition

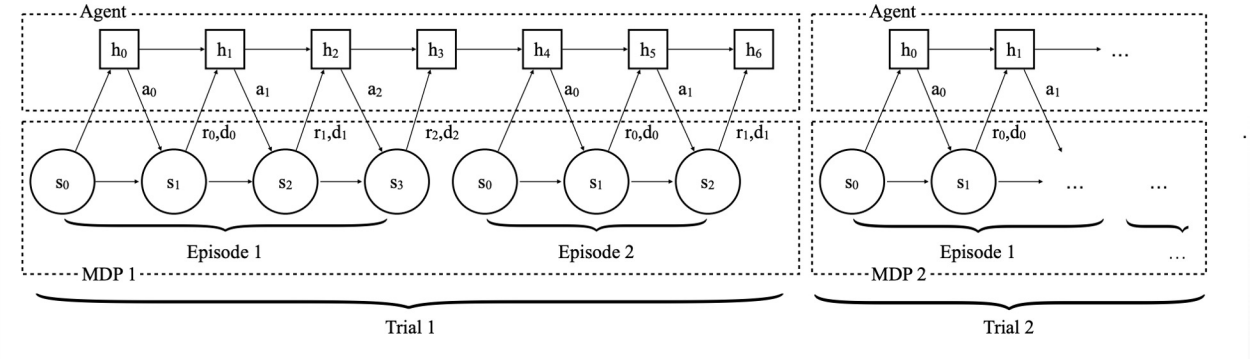
A *policy* π is a distribution over actions given states,

$$\pi(a|s) = \mathbb{P}[A_t = a \mid S_t = s]$$

- In MDP, policy depends only on current state (Markov Property)
- Policies will be time independent
- Goal of policy search method: Maximize expected discounted return

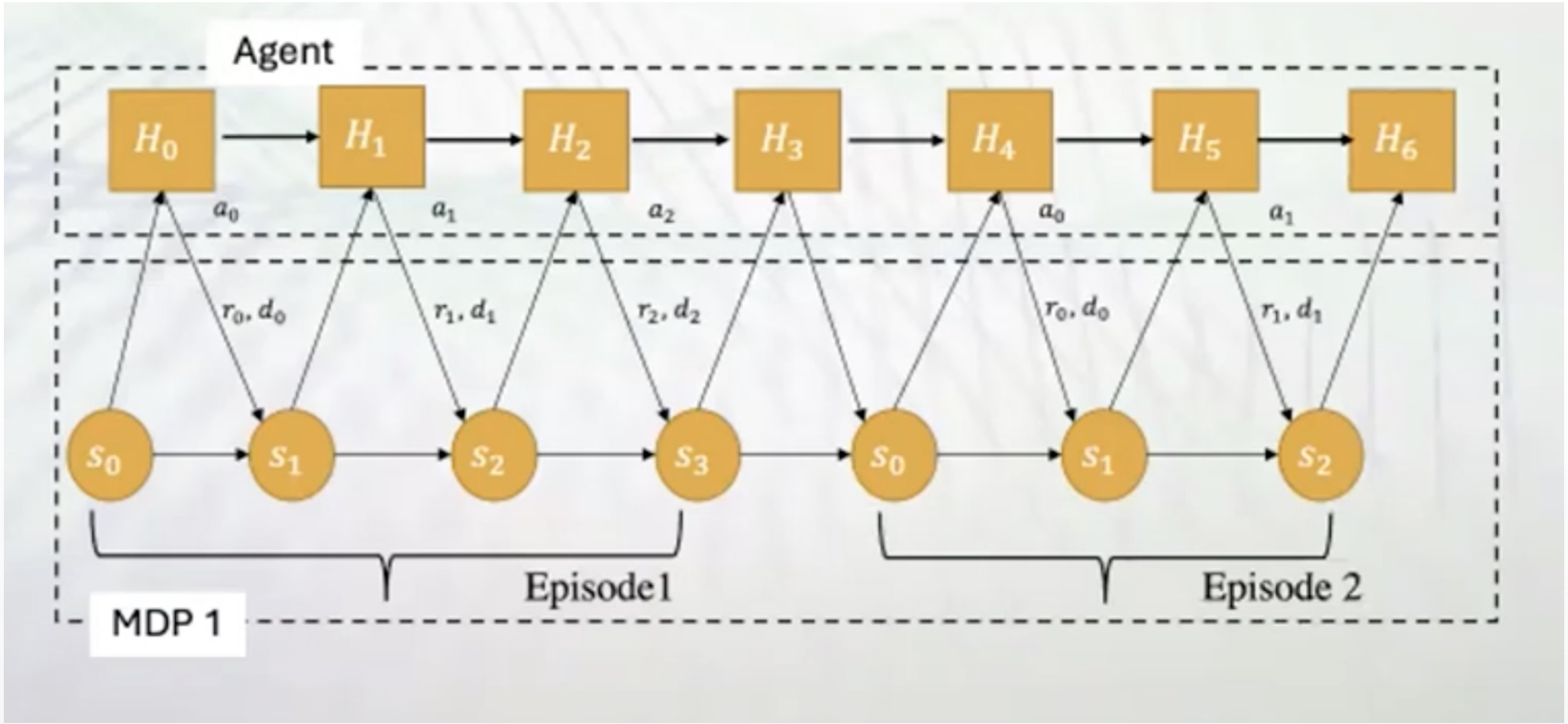
Methodology
Formulation

- Learning RL algorithm - RL problem
- Inner problem - formulated as a MDP



- Maximize expected total discounted reward averaged over a trial rather than an episode

Methodology
Formulation



Methodology - Object Interaction
Policy Representation and Optimization

- Policy - Recurrent Neural Network
- RNN uses GRUs
- Generalized Advantage Estimation (GAE) to reduce variance in stochastic gradient estimation

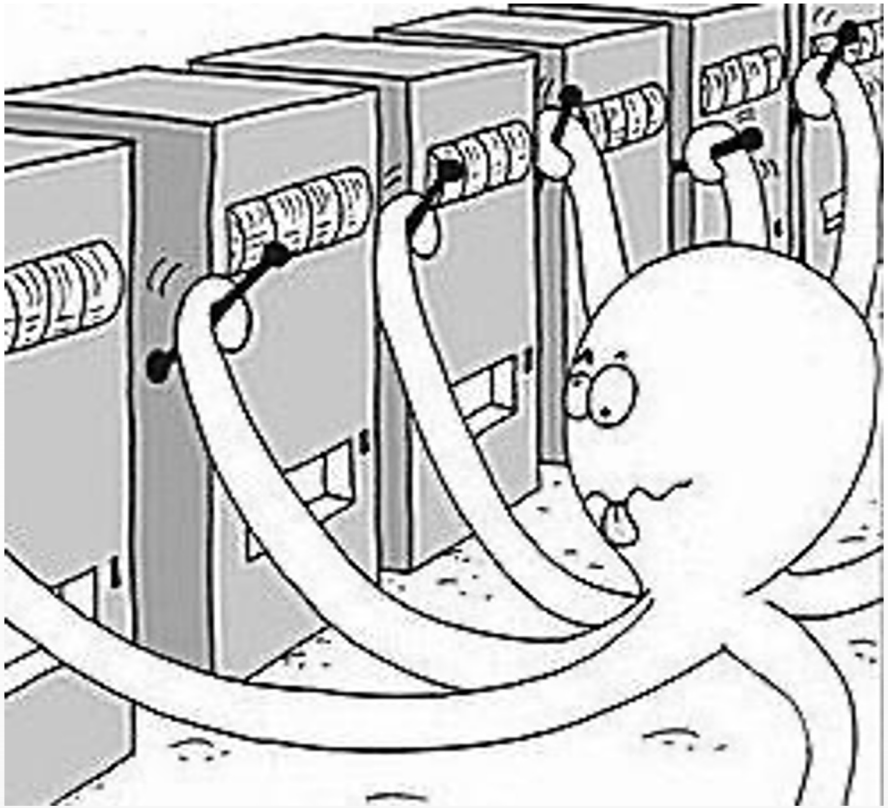
Evaluation

- Can RL^2 learn algorithms that achieve good performance on MDP classes with special structure, relative to existing algorithms tailored to this structure that have been proposed in the literature?

- 1. Multi-armed bandits (MAB)

- 2. Tabular MDPs

Evaluation
Multi-armed Bandits



https://paperswithcode.com/task/multi-armed-bandits

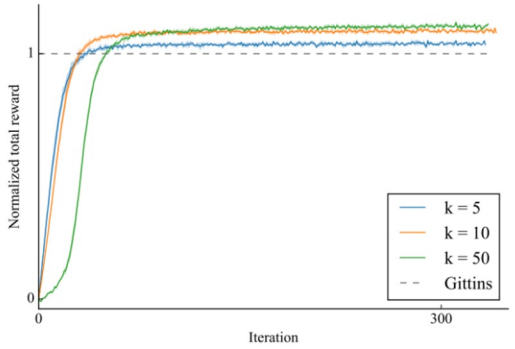
Evaluation
Multi-armed Bandits

- Subset of MDPs, agent's environment is stateless
- Goal : maximize reward over a fixed number of time steps
- Given a slot machine with n arms (bandits) with each arm having its own probability distribution of success
- Pulling any one gives reward from Bernoulli distribution
- Receive maximum rewards in a small number of steps
- Key challenge - exploring all available arms or exploiting the best arm

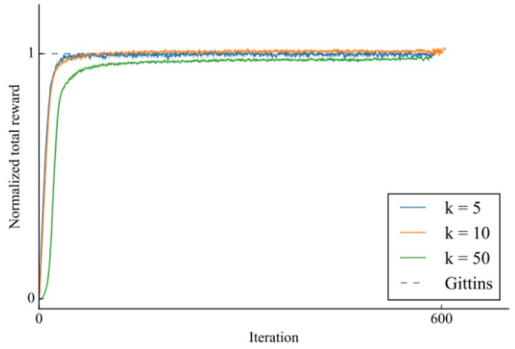
Results
Multi-armed Bandits

Setup	Random	Gittins	TS	OTS	UCB1	ϵ-Greedy	Greedy	RL^2
$n = 10, k = 5$	5.0	**6.6**	5.7	6.5	**6.7**	**6.6**	**6.6**	**6.7**
$n = 10, k = 10$	5.0	**6.6**	5.5	6.2	**6.7**	**6.6**	**6.6**	**6.7**
$n = 10, k = 50$	5.1	6.5	5.2	5.5	**6.6**	6.5	6.5	**6.8**
$n = 100, k = 5$	49.9	**78.3**	74.7	**77.9**	**78.0**	75.4	74.8	**78.7**
$n = 100, k = 10$	49.9	**82.8**	76.7	81.4	82.4	77.4	77.1	**83.5**
$n = 100, k = 50$	49.8	**85.2**	64.5	67.7	84.3	78.3	78.0	**84.9**
$n = 500, k = 5$	249.8	**405.8**	**402.0**	**406.7**	**405.8**	388.2	380.6	**401.6**
$n = 500, k = 10$	249.0	**437.8**	429.5	**438.9**	**437.1**	408.0	395.0	432.5
$n = 500, k = 50$	249.6	**463.7**	427.2	437.6	457.6	413.6	402.8	438.9

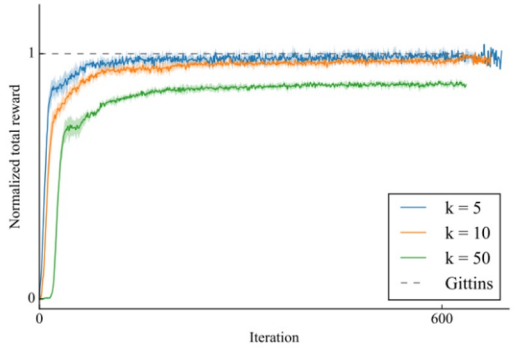
Results
Multi-armed Bandits



(a) $n = 10$



(b) $n = 100$



(c) $n = 500$

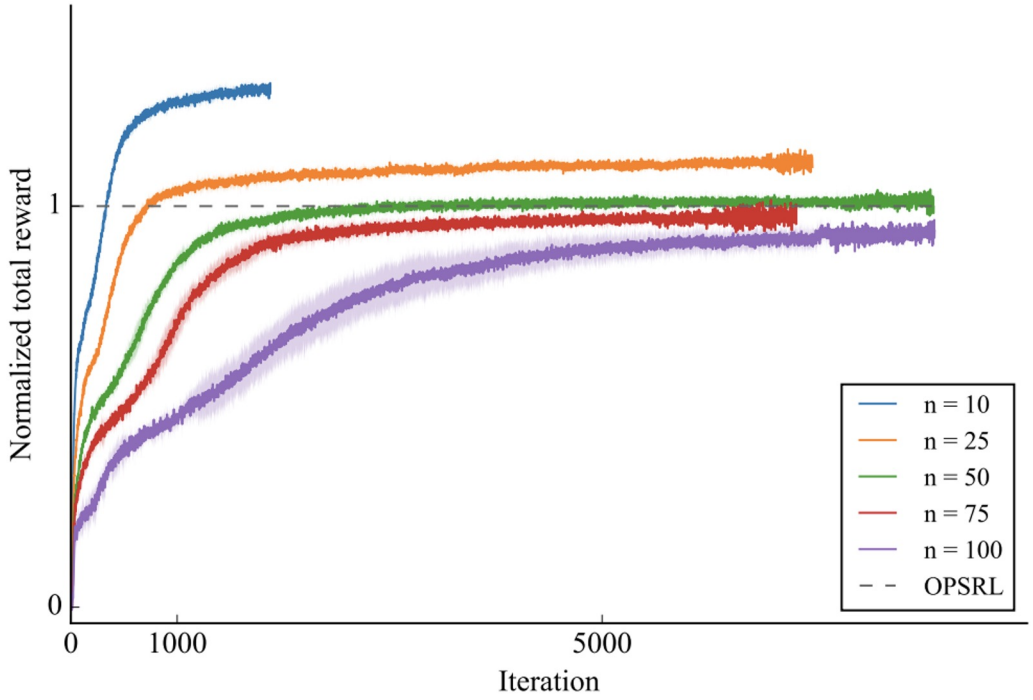
Evaluation
Tabular MDPs

- State and Action spaces are discrete and finite
- Transition probabilities and reward associated with each action is represented as a table
- Multi-armed bandit - Whether model learns to trade-off between exploration and exploitation but **no sequential decision making!**

Results
Tabular MDPs

Setup	Random	PSRL	OPSRL	UCRL2	BEB	ϵ-Greedy	Greedy	RL^2
$n = 10$	100.1	138.1	144.1	146.6	150.2	132.8	134.8	**156.2**
$n = 25$	250.2	408.8	425.2	424.1	427.8	377.3	368.8	**445.7**
$n = 50$	499.7	904.4	**930.7**	918.9	917.8	823.3	769.3	**936.1**
$n = 75$	749.9	1417.1	**1449.2**	1427.6	1422.6	1293.9	1172.9	1428.8
$n = 100$	999.4	1939.5	**1973.9**	1942.1	1935.1	1778.2	1578.5	1913.7

Results
Tabular MDPs



Evaluation Question #2

- Can RL^2 scale to high-dimensional tasks?

- Vision based navigation task

Evaluation
Vision based navigation task

- Task : Navigate a randomly generated maze to find a randomly placed target

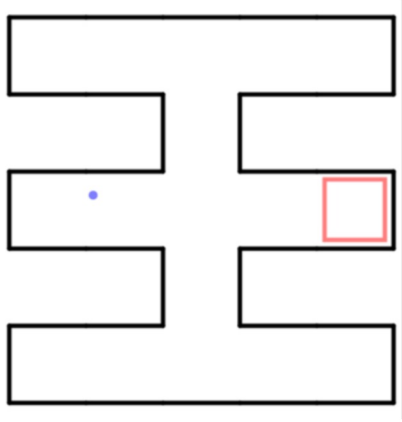
Action	Reward
Reach the target	+1
Hit the wall	-0.001
Per time step	-0.04

- Allowed to interact over multiple episodes

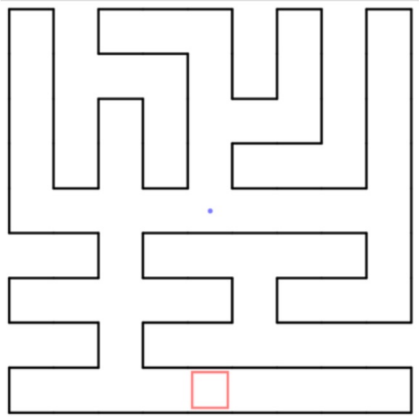
Evaluation
Vision based navigation task

- Needs to learn high level actions from scratch

(a) Sample observation



(b) Layout of the 5×5 maze in (a)



(c) Layout of a 9×9 maze

Results
Vision based navigation task

(a) Average length of successful trajectories

Episode	Small	Large
1	52.4 ± 1.3	180.1 ± 6.0
2	39.1 ± 0.9	151.8 ± 5.9
3	42.6 ± 1.0	169.3 ± 6.3
4	43.5 ± 1.1	162.3 ± 6.4
5	43.9 ± 1.1	169.3 ± 6.5

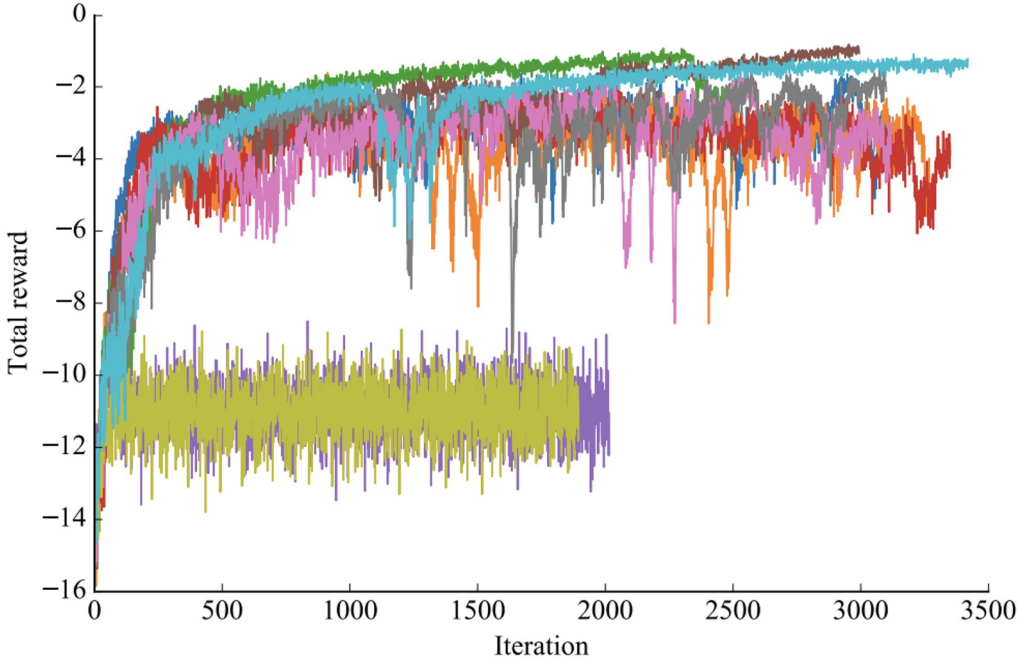
(b) %Success

Episode	Small	Large
1	99.3%	97.1%
2	99.6%	96.7%
3	99.7%	95.8%
4	99.4%	95.6%
5	99.6%	96.1%

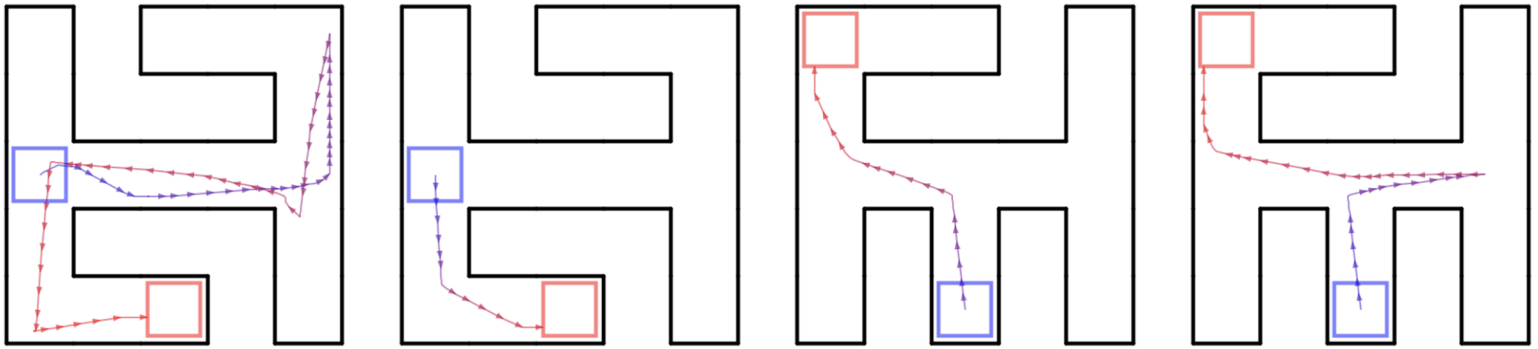
(c) %Improved

Small	Large
91.7%	71.4%

Results
Vision based navigation task



Results
Vision based navigation task



(a) Good behavior, 1st episode

(b) Good behavior, 2nd episode

(c) Bad behavior, 1st episode

(d) Bad behavior, 2nd episode

Conclusion

- Formulation : POMDP solved in outer loop where underlying MDP is unobserved by the agent
- “Fast” RL algorithm - computation whose state stored in RNN activations
- “Slow” RL algorithm - RNN’s weights are learned by this algorithm
- Comparable to theoretically optimal algorithms, scales to high dimensional tasks

Discussion

Discussion #1

- How would we apply this?
- Applications of such an algorithm in real world scenarios ?
- How do we improve the current algorithm?

Discussion #2

- **@112_f2 - Nature of memory in humans**

One thing we have not talked much about in this class is the nature of memory (with some notable exceptions). My understanding is that the human brain has structures specifically designed for short- and long-term memory, while typically in ML anything "remembered" is simply stored as fixed neural network parameters or activations when looking at a sequential input. Some models like LSTMs and GRUs attempt to address this, but suffer limitations and are often outperformed by giant transformer models.I wonder if there is still a limitation when it comes to this meta-RL in that while it may learn a good policy for learning arbitrary MDPs, the RNN structure may be inefficient when it comes to encoding good priors for other parts of the world (think "foundation models" for language, vision, etc.).

Discussion #3

- **@112_f3**

It was interesting to see how the authors applied their reinforcement learning method to a visual navigation problem with very sparse rewards. Previous papers we read that explore this problem rely on intrinsic curiosity-promoting rewards to encourage exploration and learning, while this method relies on memory (in the form of learned weights) from previous trials to improve subsequent runs in the maze. The paper points out that for the second episodes, the trajectory lengths significantly decreased because the models learned to use information from the previous episode. But I noticed that in subsequent episodes (3+) the performance seems to be worse. Does anyone know why this is the case? Or maybe I am misunderstanding how the episodes are set up?

Thank You!