Offline Evaluation of Online Reinforcement Learning Algorithms

Travis Mandel[1], Yun-En Liu[2], Emma Brunskill[3], and Zoran Popović[1,2]
[1]Center for Game Science, Computer Science & Engineering, University of Washington, Seattle, WA
[2]Enlearn™, Seattle, WA
[3]School of Computer Science, Carnegie Mellon University, Pittsburgh, PA
{tmandel, zoran}@cs.washington.edu, yunliu@enlearn.org, ebrun@cs.cmu.edu

Presentation by Revan MacQueen

Offline Evaluation of Online RL Algorithms

- ❑ Why we need offline evaluation
  - ❑ What is an evaluator?
  - ❑ Why the obvious evaluators don’t work
- ❑ Three proposed evaluation approaches
- ❑ Properties of an ideal evaluator
  - ❑ Do the proposed evaluators have these properties?
- ❑ Evaluating the evaluators: empirical results

We want to use RL in the real world

https://healthtechmagazine.net/article/2020/07/3-trends-will-influence-healthcare-staff-return-work
https://wp.nyu.edu/dispatch/2020/02/10/the-impact-of-online-education-in-academics/
https://www.theverge.com/2020/1/17/21070620/cruise-california-disengagement-report-self-driving-car

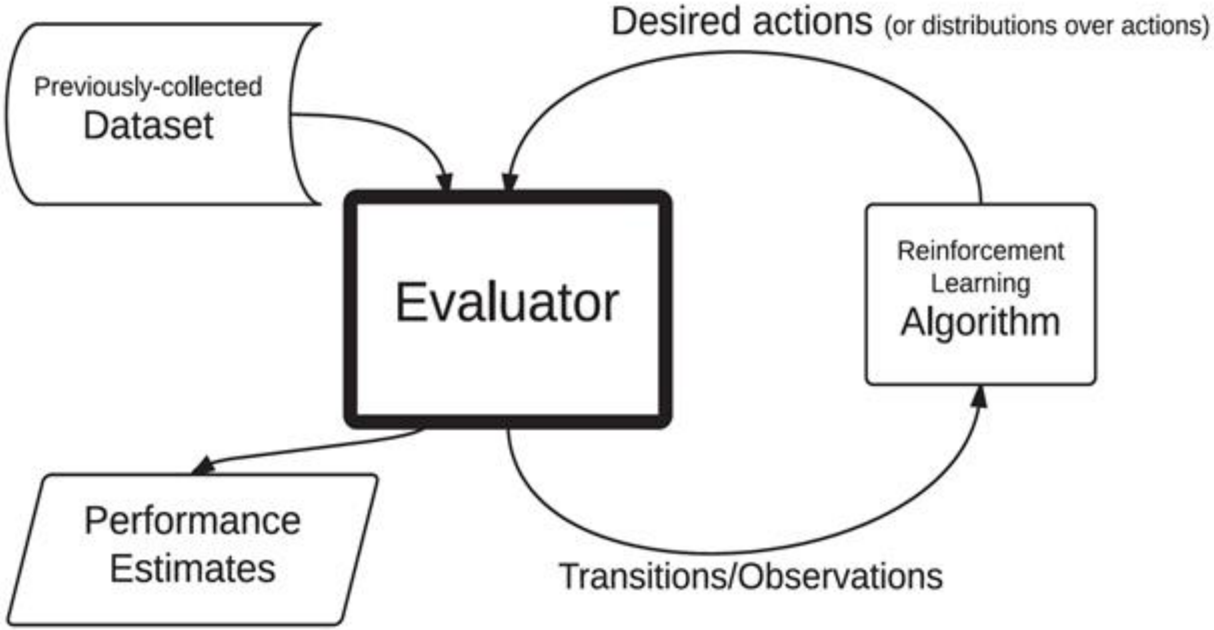
The Real World is Tricky

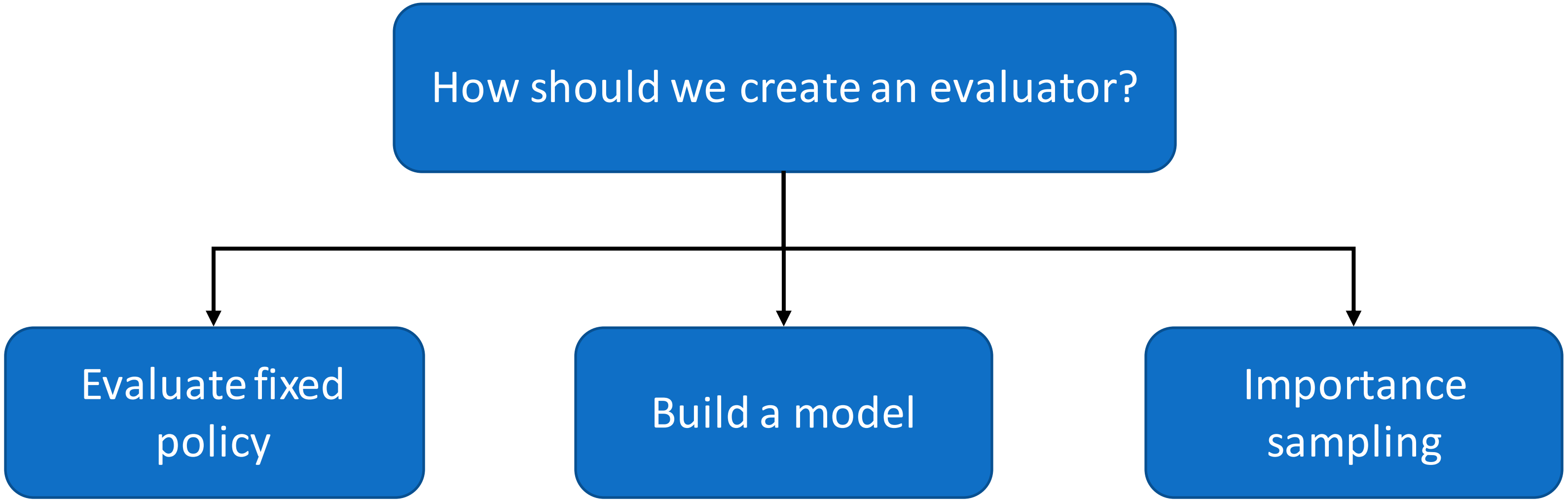
- Many applications are high risk.
- It's oftentimes computationally infeasible to try out more than one algorithm.
- Need many runs to try different hyperparameter settings.

- Ideally, we want to test learning algorithms out on real world data prior to deployment.
- We need an evaluator!

https://www.amazon.ca/Crash-Cartoon-Sticker-Decal-Design/dp/B017CZG3UG

Evaluation Overview





- Want to deploy a learning algorithm
- Can’t use a train-test split of the data.

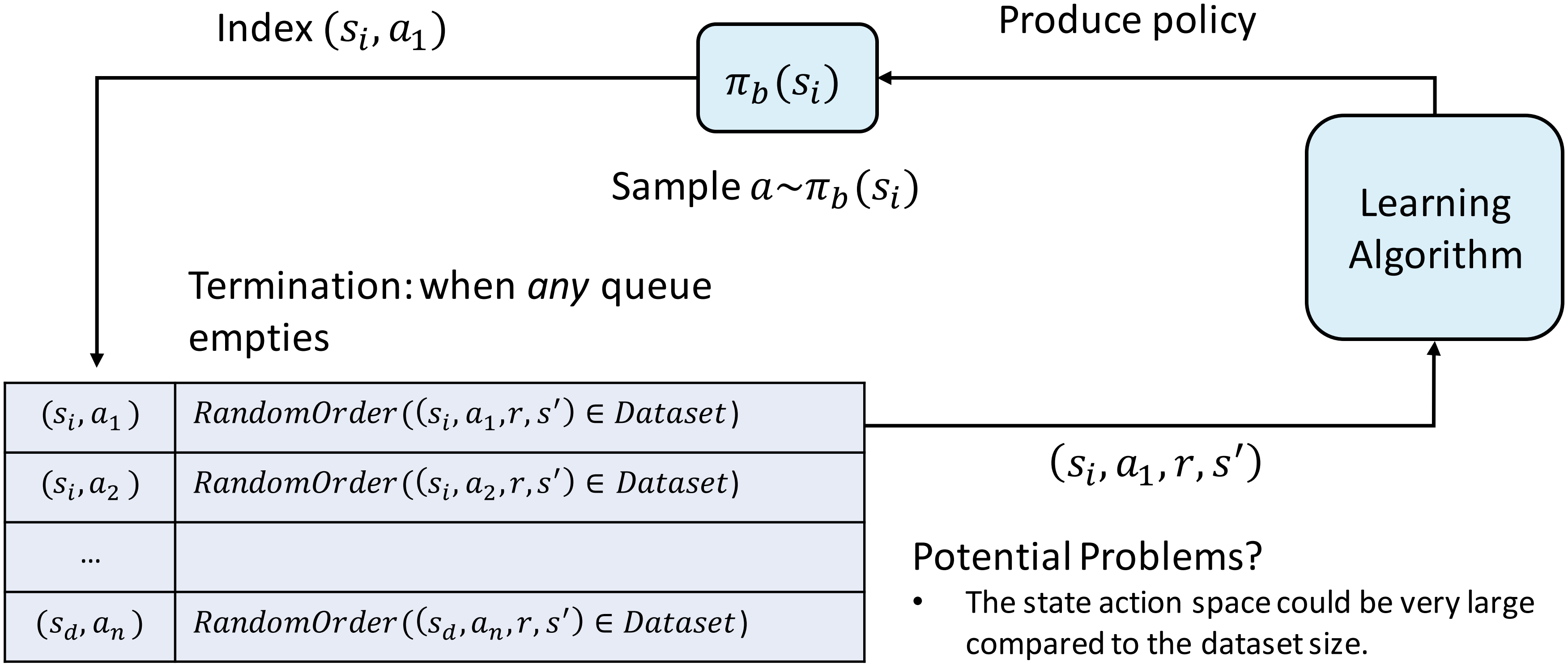
- Error accumulates at least quadratically with evaluation length (Ross, Gordon and Bagnell 2011).
- In complex problems, its unclear how to build such a model.

- Variance is unusably high if algorithm is evaluated for hundreds of time steps (Dudík et al. 2014).

Offline Evaluation of Online RL Algorithms

- ☑ Why we need offline evaluation
  - ☑ What is an evaluator?
  - ☑ Why the obvious evaluators don't work
- ❑ **Three proposed evaluation approaches in this work**
- ❑ Properties of an ideal evaluator
  - ❑ Do the proposed evaluators have these properties?
- ❑ Evaluating the evaluators: empirical results

Evaluator 1: Queue-based Evaluator



(s_i, a_1)	$RandomOrder((s_i, a_1, r, s') \in Dataset)$
(s_i, a_2)	$RandomOrder((s_i, a_2, r, s') \in Dataset)$
…	
(s_d, a_n)	$RandomOrder((s_d, a_n, r, s') \in Dataset)$

Potential Problems?

- The state action space could be very large compared to the dataset size.
- Could result in early termination!

Leveraging Policy Similarity

- The Queue-based evaluator suffers from sample inefficiency.

(s, a_1)	(s, a_1, r, s')
(s, a_2)	$(s, a_2, r, s'), (s, a_2, r, s'), \ldots, (s, a_2, r, s')$

- If a_1 is ever sampled, Queue-based must terminate in the next iteration.
- If we know the sampling distribution, we can do a lot better!

What if the learned policy is the same as the sampling distribution?

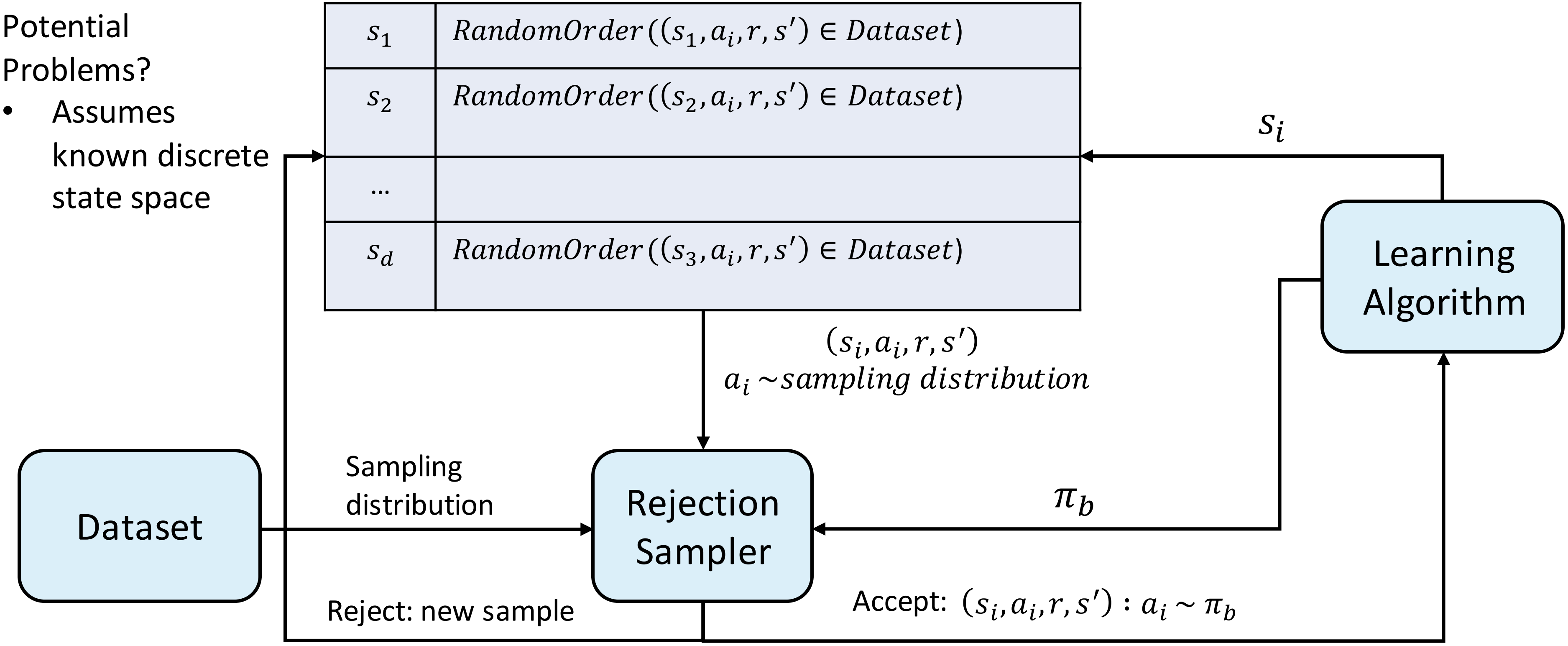
Potentially all the data can be used!

Revealed randomness

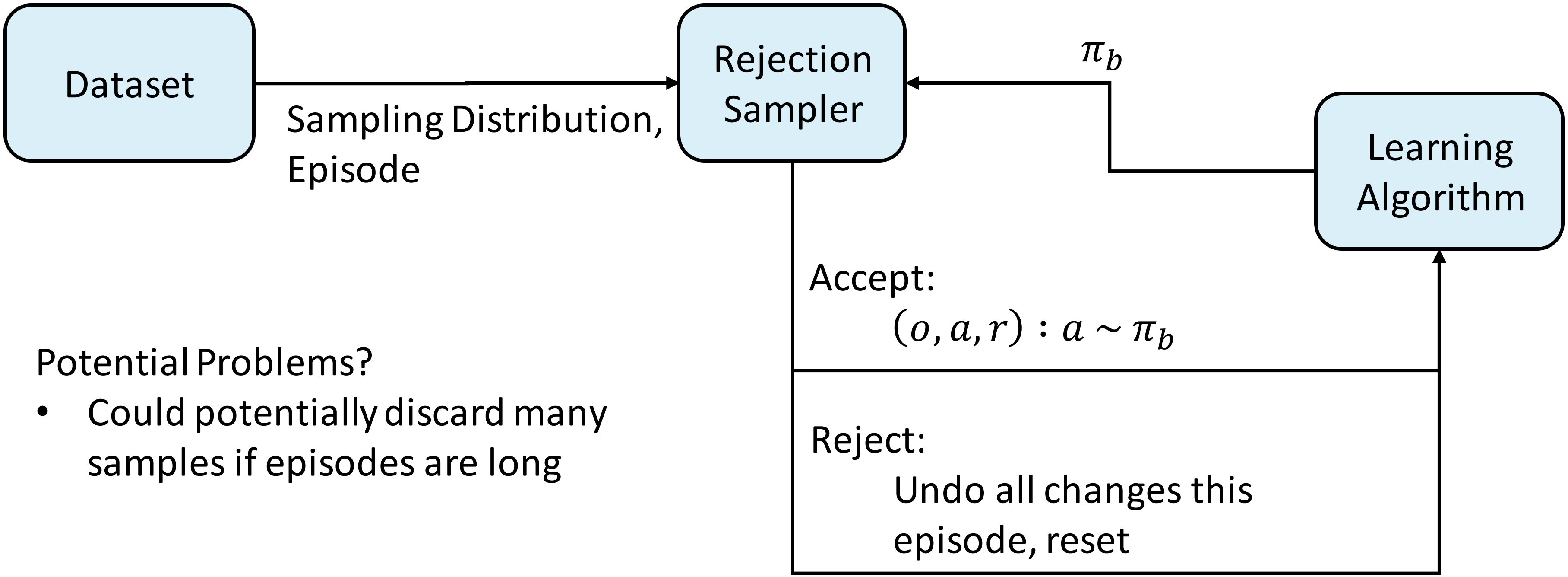
Evaluator 2: Per State Rejection Sampling (PSRS)

Potential Problems?

- Assumes known discrete state space



Evaluator 3: Per Episode Rejection Sampling (PERS)



Potential Problems?

- Could potentially discard many samples if episodes are long

Summary of Evaluators

- Queue-based
  - 💡 Keep a queue for all (s,a) pairs and store (r,s')
  - ☹ Data inefficiency
- Per State Rejection Sampling
  - 💡 Use rejection sampling to sample transitions from dataset
  - ☹ Assumes known state space
- Per Episode Rejection Sampling
  - 💡 Eliminate reliance on known and discrete state by sampling entire episodes
  - ☹ Could potentially discard many samples if episodes are long

Offline Evaluation of Online RL Algorithms

- ☑ Why we need offline evaluation
  - ☑ What is an evaluator?
  - ☑ Why the obvious evaluators don't work
- ☑ Three proposed evaluation approaches in this work
- ❑ **Properties of an ideal evaluator.**
  - ❑ Do the proposed evaluators have these properties?
- ❑ Evaluating the evaluators: empirical results

6 Properties Of An Ideal Evaluator

1. *(s,a,r,s')* tuples provided to algorithm have the same distribution as the true MDP.

   Queue ✓ PSRS ✓ PERS ✓

2. High sample efficiency.

   "domain dependent"

3. Evaluator returns unbiased performance estimates.

   Queue ✗ PSRS ✗ PERS ✗

   PERS variant ✓

4. Evaluator can use data from an unknown sampling distribution.

   Queue ✓ PSRS ✗ PERS ✗

5. Does not assume environment is a discrete MDP.

   Queue ✗ PSRS ✗ PERS ✓

6. Computationally efficient.

   Queue ✓ PSRS ✓ PERS ?

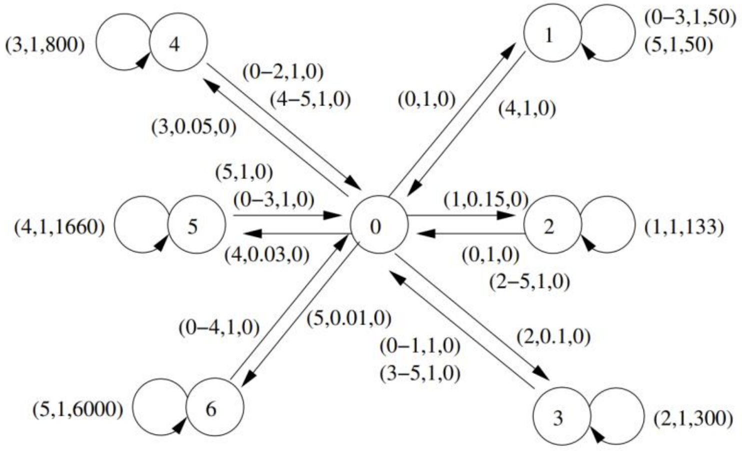
Offline Evaluation of Online RL Algorithms

- ☑ Why we need offline evaluation
  - ☑ What is an evaluator?
  - ☑ Why the obvious evaluators don't work
- ☑ Three proposed evaluation approaches in this work
- ☑ Properties of an ideal evaluator
  - ☑ Do the proposed evaluators have these properties?
- ❑ **Evaluating the evaluators: empirical results**

Experiment 1: Six Arms

- Accuracy of PSRS vs. a model-based approach
- Model-based approach uses dataset to build MLE model.

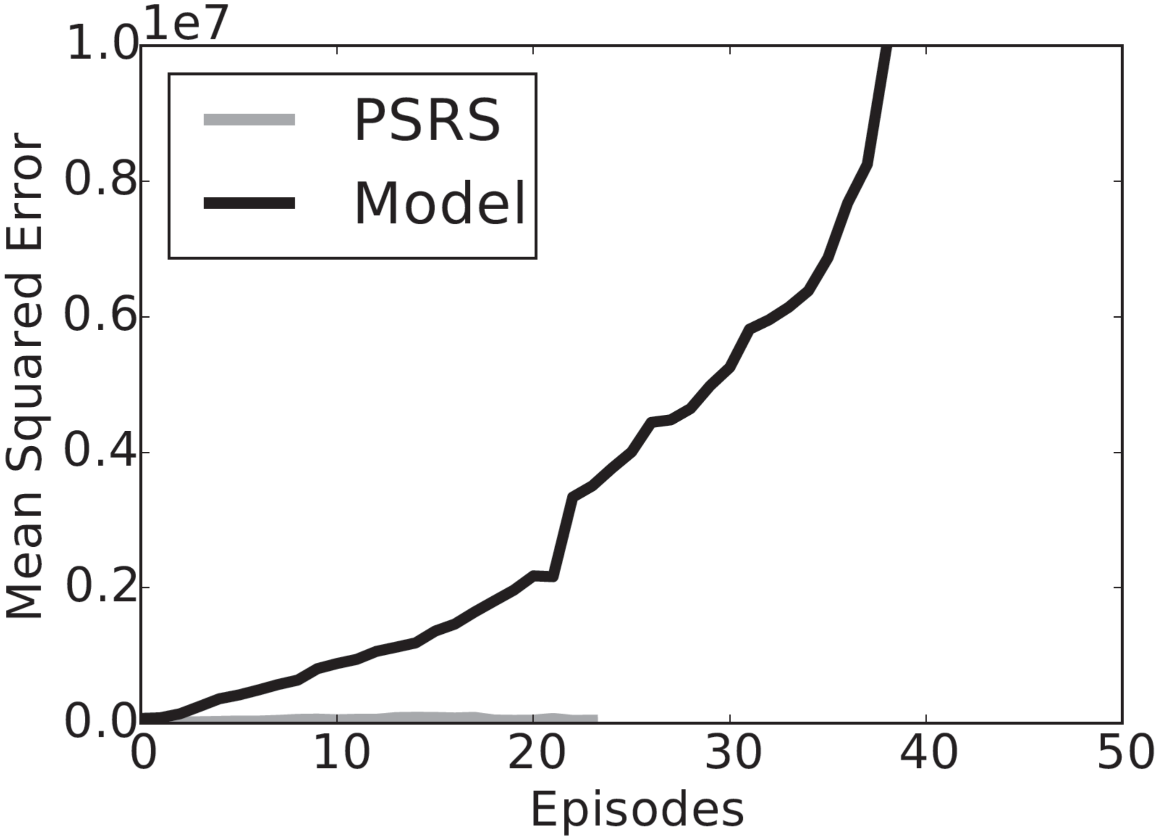
Learning algorithm	Posterior Sampling Reinforcement Learning (PSRL), 10 posterior samples
Dataset	100 datasets of 100 samples
Sampling distribution	Uniform
Performance estimate	Cumulative reward
Number of runs per dataset	10



SixArms

PSRS vs Model-based on Six Arms

- 1000 runs
- MSE is over estimated return vs true return on environment

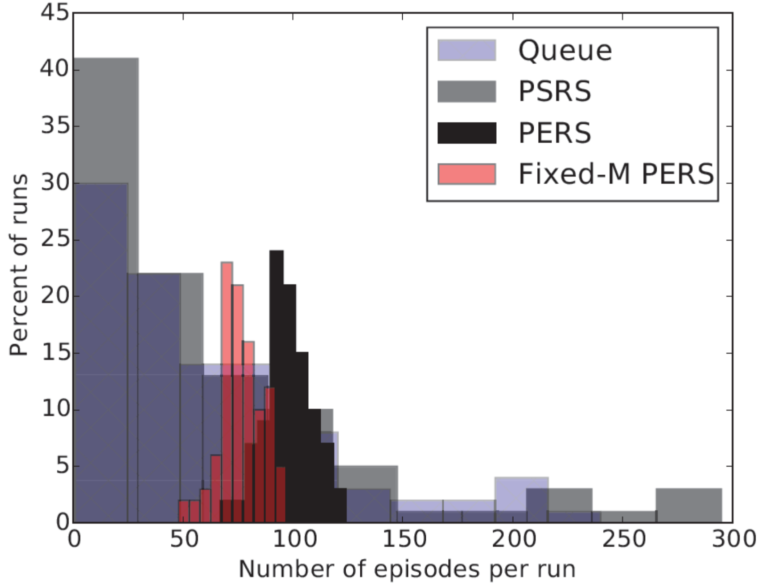


Experiment 2: Treefrog Treasure

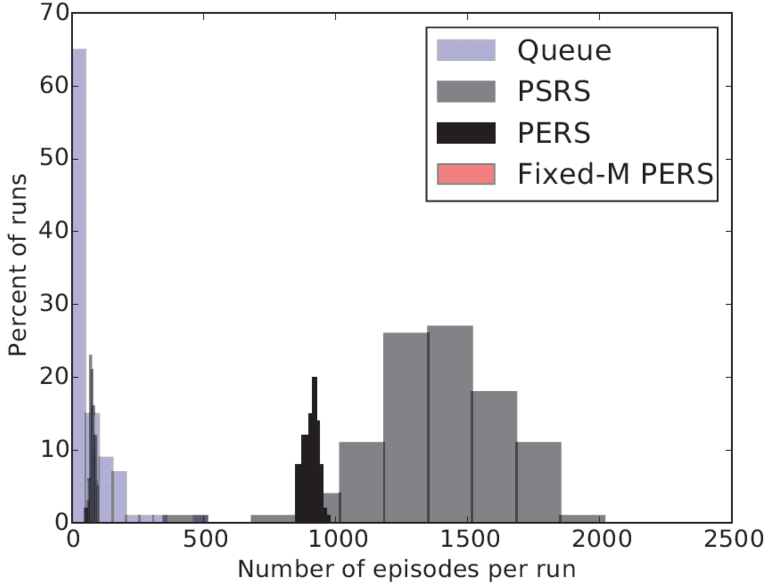
- Evaluating sample efficiency

Learning algorithm	Posterior Sampling Reinforcement Learning (PSRL)
Dataset	Episodes of 11,550 players
Sampling distribution	Semi-uniform
Number of runs per dataset	100
Horizon	3 time steps

Performance of Evaluators on Treefrog Treasure



1 PSRL posterior sample



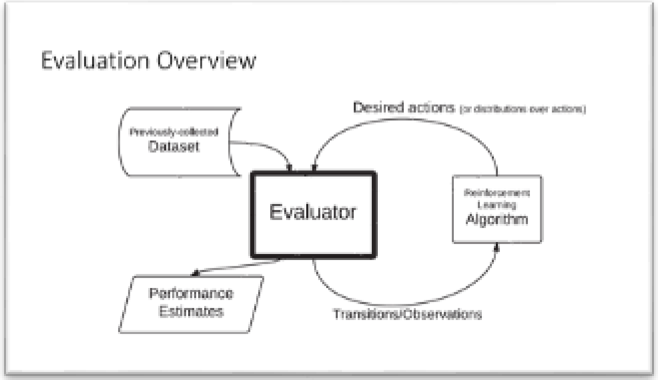
10 PSRL posterior samples
(more revealed randomness)

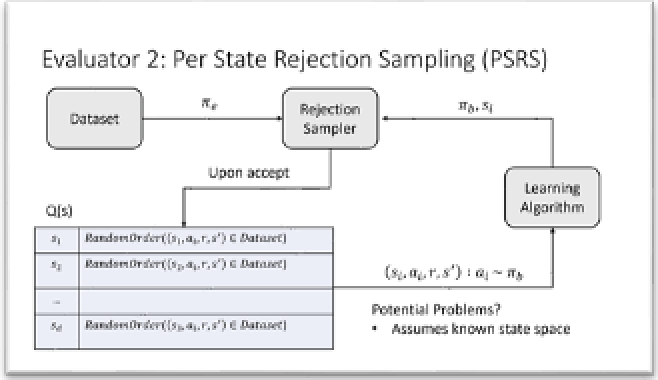
Quick Thoughts On The Evaluation

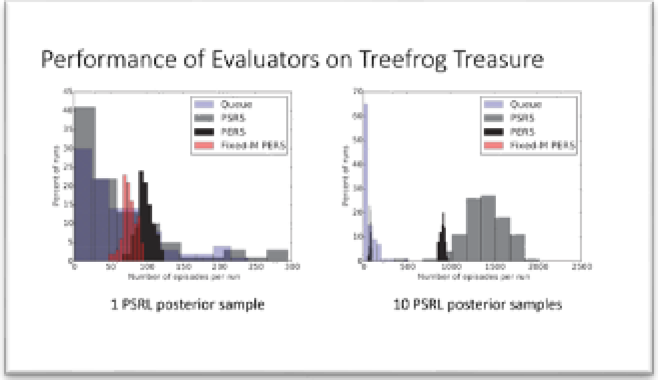
- Why did they choose these particular environments?
- Why were Queue-based and PERS omitted from the Six Arms experiment?
- Why didn't they test fixed policy evaluators?
- Why didn't they test importance sampling-based approaches?

Conclusion


We want to use RL in the real world


Evaluation Overview
Desired actions
Previously-collected Dataset
Evaluator
Reinforcement Learning Algorithm
Performance Estimates
Transitions/Observations


Evaluator 2: Per State Rejection Sampling (PSRS)
Dataset
π_e
Rejection Sampler
π_b, s_i
Upon accept
Learning Algorithm
Q(s)
$(s_i, a_i, r, s') : a_i \sim \pi_b$
Potential Problems?
• Assumes known state space


Performance of Evaluators on Treefrog Treasure
1 PSRL posterior sample
10 PSRL posterior samples