



Computer Engineering Department

# Exploration in RL

**Mohammad Hossein Rohban, Ph.D.**

Spring 2025

Courtesy: Most of slides are adopted from CS 285, UC Berkeley.

# What's the problem?

this is easy (mostly)



Why?

this is impossible



# Montezuma's revenge



- Getting key = reward
- Opening door = reward
- Getting killed by skull = nothing (is it good? bad?)
- Finishing the game only weakly correlates with rewarding events
- We know what to do because we understand what these sprites mean!

# Why exploration can be difficult?

- **Temporally extended** tasks like Montezuma's revenge become increasingly difficult based on
  - How extended the task is
  - How little you know about the rules
- Let's assume a **complex task**
  - Consisting of **multiple sub-task, each is a prerequisite for the next sub-task.**
  - Each should be solved **in a sequence** to get a high reward
  - Epsilon greedy does not obviously help:
  - Suppose you mastered up to the  $k^{\text{th}}$  sub-task.
  - You have to **exploit up to the  $k^{\text{th}}$  task** and then **explore onwards.**
  - Now the chance to only explore in the sub-task  $(k+1)$  is  $(1 - \epsilon)^{O(k)} \epsilon^{O(1)}$ .
  - For  $\epsilon = 0.1$ ,  $k = 5$ , this is  $\sim 6\%$ . For  $\epsilon = 0.5$ , this is  $\sim 3\%$ .

# Exploration and exploitation

- Two potential definitions of exploration problem:
  - How can an agent **discover high-reward strategies** that require a temporally **extended sequence of complex behaviors** that, **individually, are not rewarding**?
  - How can an agent decide whether to **attempt new behaviors** (to discover ones with higher reward) or **continue to do the best thing it knows so far**?

# Optimal Exploration?

- Bayesian model of the environment. (POMDP with belief state)
- Optimize the expected reward under **all uncertainties**.
- Requires knowledge of **state dynamic distribution class**, the **prior**, and **maintaining the belief state**.
- Here we seek **simpler** solutions which could be extended to more **complex** scenarios.
- Compare the regret in such models against the Bayes' optimal approach.

$$\text{Reg}(T) = TE[r(a^*)] - \sum_{t=1}^T r(a_t)$$

expected reward of best action  
(the best we can hope for in expectation)      actual reward of action  
actually taken

# Bandits

assume  $r(a_i) \sim p_{\theta_i}(r_i)$

e.g.,  $p(r_i = 1) = \theta_i$  and  $p(r_i = 0) = 1 - \theta_i$

$\theta_i \sim p(\theta)$ , but otherwise unknown

this defines a POMDP with  $\mathbf{s} = [\theta_1, \dots, \theta_n]$

belief state is  $\hat{p}(\theta_1, \dots, \theta_n)$

how do we measure goodness of exploration algorithm?

regret: difference from optimal policy at time step  $T$ :

$$\text{Reg}(T) = \underbrace{TE[r(a^*)]}_{\substack{\text{expected reward of best action} \\ \text{(the best we can hope for in expectation)}}} - \sum_{t=1}^T \underbrace{r(a_t)}_{\substack{\text{actual reward of action} \\ \text{actually taken}}}$$

- solving the POMDP yields the optimal exploration strategy
- but that's overkill: belief state is huge!
- we can do very well with much simpler strategies

# Optimistic exploration

keep track of average reward  $\hat{\mu}_a$  for each action  $a$

exploitation: pick  $a = \arg \max \hat{\mu}_a$

optimistic estimate:  $a = \arg \max \hat{\mu}_a + \underbrace{C \sigma_a}_{\text{some sort of variance estimate}}$

**intuition: try each arm until you are *sure* it's not great**

example (Auer et al. Finite-time analysis of the multiarmed bandit problem):

$$a = \arg \max \hat{\mu}_a + \sqrt{\frac{2 \ln T}{N(a)}} \quad \leftarrow \begin{array}{l} \text{number of times we} \\ \text{picked this action} \end{array}$$

$\text{Reg}(T)$  is  $O(\log T)$ , provably as good as any algorithm




# Probability matching/posterior sampling

assume  $r(a_i) \sim p_{\theta_i}(r_i)$

this defines a POMDP with  $\mathbf{s} = [\theta_1, \dots, \theta_n]$

belief state is  $\hat{p}(\theta_1, \dots, \theta_n)$

this is a *model* of our bandit



idea: sample  $\theta_1, \dots, \theta_n \sim \hat{p}(\theta_1, \dots, \theta_n)$

pretend the model  $\theta_1, \dots, \theta_n$  is correct

take the optimal action

update the model

- This is called posterior sampling or Thompson sampling
- Harder to analyze theoretically
- Can work very well empirically
- See: Chapelle & Li, “An Empirical Evaluation of Thompson Sampling.”

# Information gain

## Bayesian experimental design:

say we want to determine some latent variable  $z$  (e.g.,  $z$  might be the optimal action, or its value)

which action do we take?

let  $\mathcal{H}(\hat{p}(z))$  be the current entropy of our  $z$  estimate

let  $\mathcal{H}(\hat{p}(z)|y)$  be the entropy of our  $z$  estimate after observation  $y$  (e.g.,  $y$  might be  $r(a)$ )

the lower the entropy, the more precisely we know  $z$

$$\text{IG}(z, y) = E_y[\mathcal{H}(\hat{p}(z)) - \mathcal{H}(\hat{p}(z)|y)]$$

typically depends on action, so we have  $\text{IG}(z, y|a)$

# Information gain example

$$\text{IG}(z, y|a) = E_y[\mathcal{H}(\hat{p}(z)) - \mathcal{H}(\hat{p}(z)|y)|a]$$

how much we learn about  $z$  from action  $a$ , given current beliefs

Example bandit algorithm:

Russo & Van Roy “Learning to Optimize via Information-Directed Sampling”

$y = r_a$ ,  $z = \theta_a$  (parameters of model  $p(r_a)$ )

$g(a) = \text{IG}(\theta_a, r_a|a)$  – information gain of  $a$

$\Delta(a) = E[r(a^*) - r(a)]$  – expected suboptimality of  $a$

choose  $a$  according to  $\arg \min_a \frac{\Delta(a)^2}{g(a)}$

← don't take actions that you're sure are suboptimal

↙ don't bother taking actions if you won't learn anything

# General themes

UCB:

$$a = \arg \max \hat{\mu}_a + \sqrt{\frac{2 \ln T}{N(a)}}$$

Thompson sampling:

$$\theta_1, \dots, \theta_n \sim \hat{p}(\theta_1, \dots, \theta_n)$$

$$a = \arg \max_a E_{\theta_a}[r(a)]$$

Info gain:

$$\text{IG}(z, y|a)$$

- Most exploration strategies require some kind of uncertainty estimation (even if it's naïve)
- Usually assumes some value to new information
  - Assume unknown = good (optimism)
  - Assume sample = truth
  - Assume information gain = good

# Optimistic exploration in RL

$$\text{UCB: } a = \arg \max \hat{\mu}_a + \underbrace{\sqrt{\frac{2 \ln T}{N(a)}}}_{\text{"exploration bonus"}}$$

lots of functions work, so long as they decrease with  $N(a)$

can we use this idea with MDPs?

count-based exploration: use  $N(\mathbf{s}, \mathbf{a})$  or  $N(\mathbf{s})$  to add *exploration bonus*

use  $r^+(\mathbf{s}, \mathbf{a}) = r(\mathbf{s}, \mathbf{a}) + \mathcal{B}(N(\mathbf{s}))$



bonus that decreases with  $N(\mathbf{s})$

use  $r^+(\mathbf{s}, \mathbf{a})$  instead of  $r(\mathbf{s}, \mathbf{a})$  with any model-free algorithm

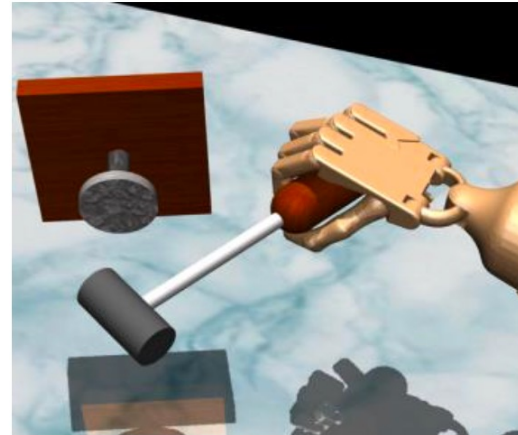
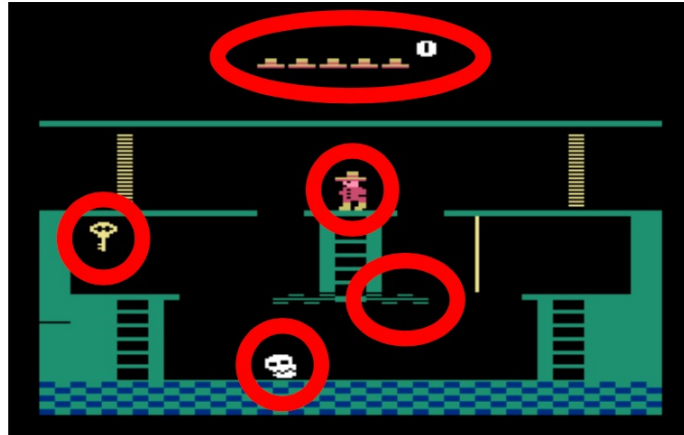
+ simple addition to any RL algorithm

- need to tune bonus weight

# The trouble with counts

$$\text{use } r^+(\mathbf{s}, \mathbf{a}) = r(\mathbf{s}, \mathbf{a}) + \mathcal{B}(N(\mathbf{s}))$$

But wait... what's a count?



Uh oh... we never see the same thing twice!  
But some states are more similar than others

# Fitting generative models

idea: fit a density model  $p_{\theta}(\mathbf{s})$  (or  $p_{\theta}(\mathbf{s}, \mathbf{a})$ )

$p_{\theta}(\mathbf{s})$  might be high even for a new  $\mathbf{s}$

if  $\mathbf{s}$  is similar to previously seen states

can we use  $p_{\theta}(\mathbf{s})$  to get a “pseudo-count”?

if we have small MDPs

the true probability is:

after we see  $\mathbf{s}$ , we have:

$$P(\mathbf{s}) = \frac{N(\mathbf{s})}{n}$$


probability/density

count

total states visited

$$P'(\mathbf{s}) = \frac{N(\mathbf{s}) + 1}{n + 1}$$

# Exploring with pseudo-counts



- fit model  $p_\theta(\mathbf{s})$  to all states  $\mathcal{D}$  seen so far
- take a step  $i$  and observe  $\mathbf{s}_i$
- fit new model  $p_{\theta'}(\mathbf{s})$  to  $\mathcal{D} \cup \mathbf{s}_i$
- use  $p_\theta(\mathbf{s}_i)$  and  $p_{\theta'}(\mathbf{s}_i)$  to estimate  $\hat{N}(\mathbf{s})$
- set  $r_i^+ = r_i + \mathcal{B}(\hat{N}(\mathbf{s}))$  ← “pseudo-count”

how to get  $\hat{N}(\mathbf{s})$ ? use the equations

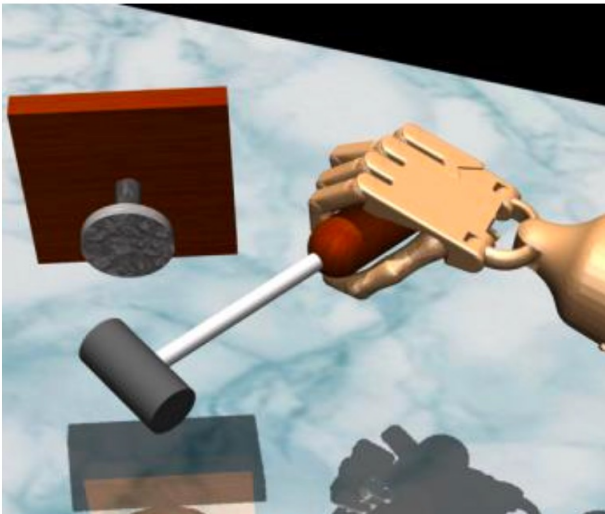
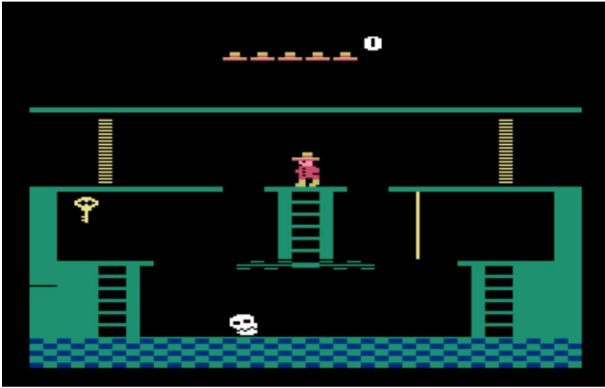
$$p_\theta(\mathbf{s}_i) = \frac{\hat{N}(\mathbf{s}_i)}{\hat{n}} \qquad p_{\theta'}(\mathbf{s}_i) = \frac{\hat{N}(\mathbf{s}_i) + 1}{\hat{n} + 1}$$

two equations and two unknowns!

$$\hat{N}(\mathbf{s}_i) = \hat{n} p_\theta(\mathbf{s}_i) \qquad \hat{n} = \frac{1 - p_{\theta'}(\mathbf{s}_i)}{p_{\theta'}(\mathbf{s}_i) - p_\theta(\mathbf{s}_i)} p_\theta(\mathbf{s}_i)$$



# What kind of generative modeling to use?

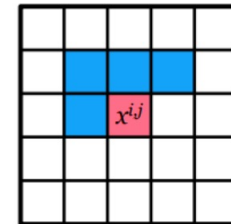


$$p_{\theta}(\mathbf{s})$$

need to be able to output densities, but doesn't necessarily need to produce great samples

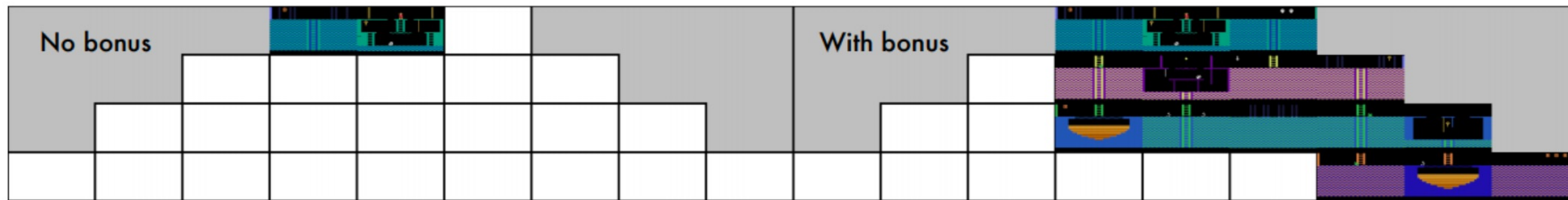
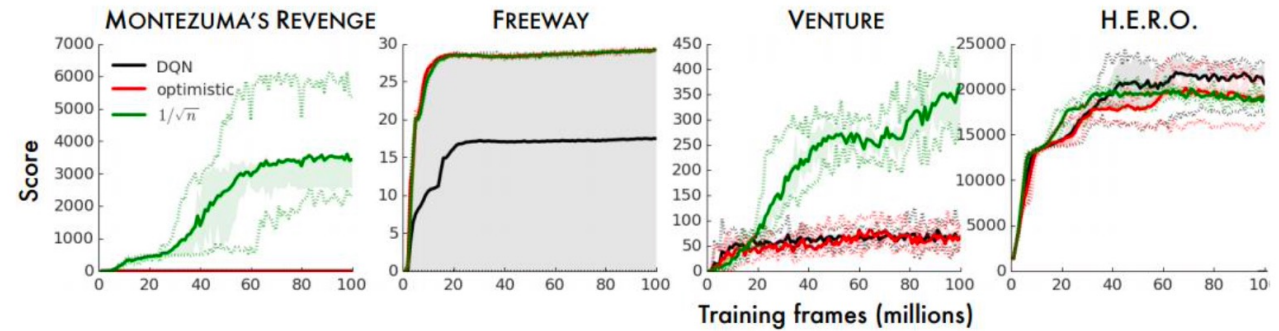
opposite considerations from many popular generative models in the literature (e.g., GANs)

Bellemare et al.: "CTS" model:  
condition each pixel on its top-left neighborhood



Other models: stochastic neural networks, compression length, EX2

# Does it work?



Bellemare et al. "Unifying Count-Based Exploration..."

# Counting with hashes

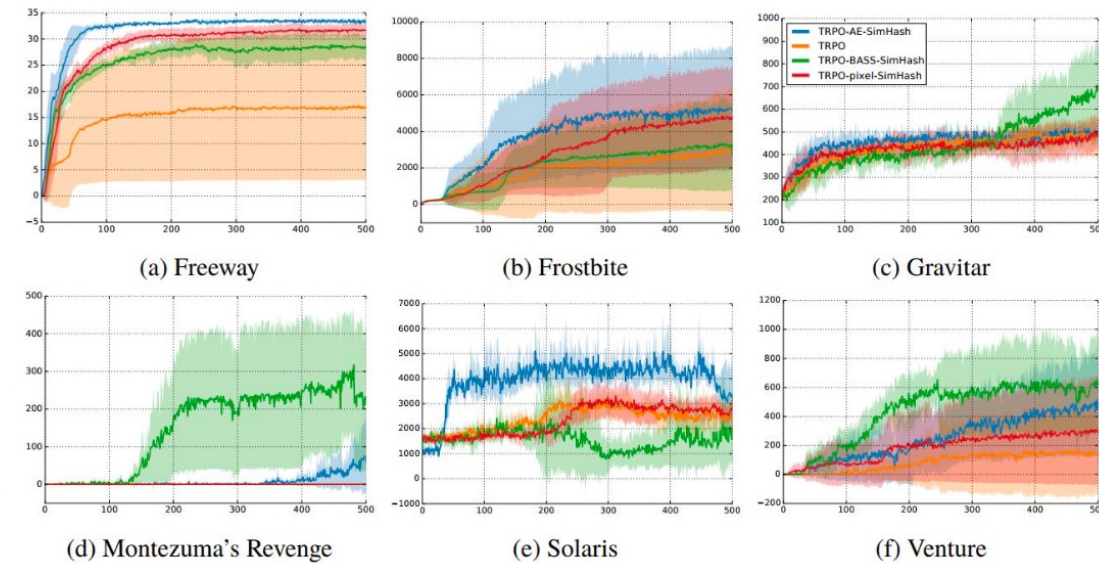
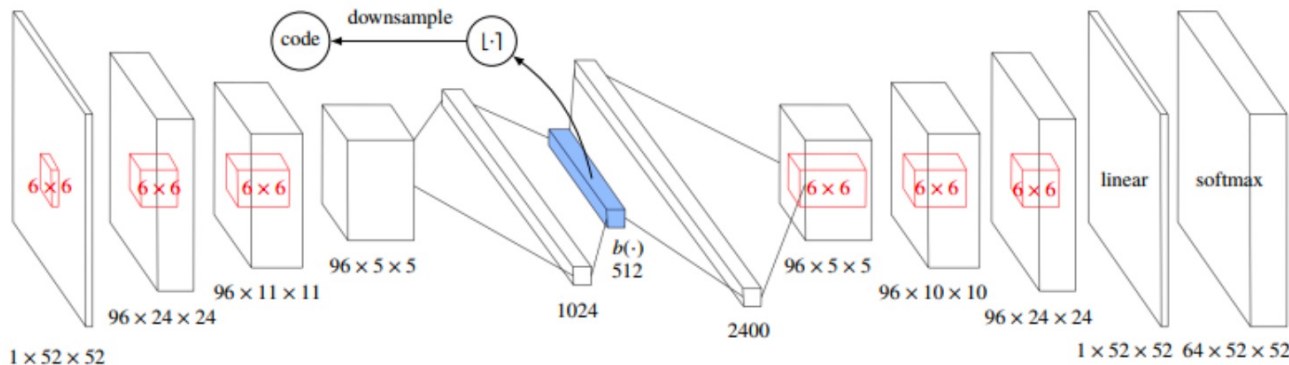
What if we still count states, but in a different space?

idea: compress  $s$  into a  $k$ -bit code via  $\phi(s)$ , then count  $N(\phi(s))$

shorter codes = more hash collisions

similar states get the same hash? maybe

improve the odds by *learning* a compression:



# Implicit density modeling with exemplar models

$p_{\theta}(\mathbf{s})$       need to be able to output densities, but doesn't  
necessarily need to produce great samples

Can we explicitly compare the new state to past states?

Intuition: the state is **novel** if it is **easy** to distinguish from all  
previous seen states by a classifier

for each observed state  $\mathbf{s}$ , fit a classifier to classify that state against all past  
states  $\mathcal{D}$ , use classifier error to obtain density

$$p_{\theta}(\mathbf{s}) = \frac{1 - D_{\mathbf{s}}(\mathbf{s})}{D_{\mathbf{s}}(\mathbf{s})}$$

← probability that classifier assigns that  $\mathbf{s}$  is “positive”  
positives:  $\{\mathbf{s}\}$   
negatives:  $\mathcal{D}$

# Implicit density modeling with exemplar models

$$D_{x^*} = \arg \max_{D \in \mathcal{D}} (E_{\delta_{x^*}} [\log D(x)] + E_{P_{\mathcal{X}}} [\log 1 - D(x)]) . \quad (1)$$

**Proposition 1.** (*Optimal Discriminator*) For a discrete distribution  $P_{\mathcal{X}}(x)$ , the optimal discriminator  $D_{x^*}$  for exemplar  $x^*$  satisfies

$$D_{x^*}(x) = \frac{\delta_{x^*}(x)}{\delta_{x^*}(x) + P_{\mathcal{X}}(x)} \quad \text{and} \quad D_{x^*}(x^*) = \frac{1}{1 + P_{\mathcal{X}}(x^*)} .$$


*Proof.* The proof is obtained by taking the derivative of the loss in Eq. (1) with respect to  $D(x)$ , setting it to zero, and solving for  $D(x)$ .  $\square$



# Implicit density modeling with exemplar models

hang on... aren't we just checking if  $s = s$ ?

if  $s \in \mathcal{D}$ , then the optimal  $D_s(s) \neq 1$

in fact:  $D_s^*(s) = \frac{1}{1 + p(s)}$    $p_\theta(s) = \frac{1 - D_s(s)}{D_s(s)}$

in reality, each state is unique, so we *regularize* the classifier

isn't one classifier per state a bit much?

train one *amortized* model: single network that takes in exemplar as input!

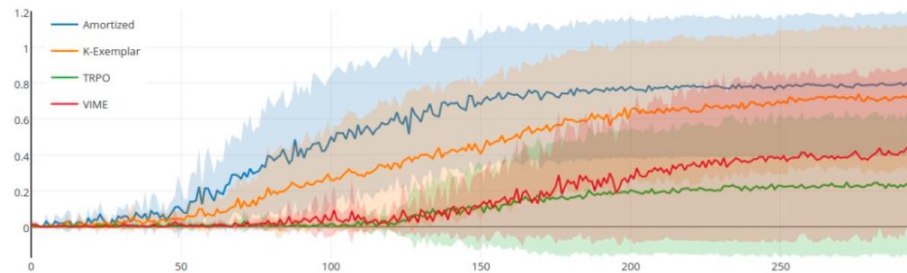
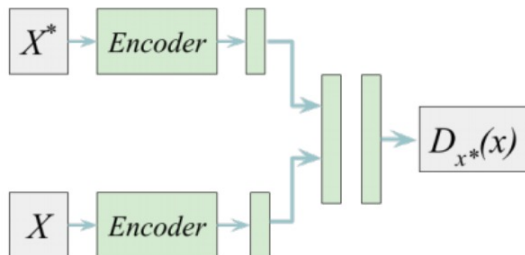


Figure 9: DoomMyWayHome+

Fu et al. "EX2: Exploration with Exemplar Models..."

# Heuristic estimation of counts via errors

$p_\theta(\mathbf{s})$

need to be able to output densities, but doesn't necessarily need to produce great samples

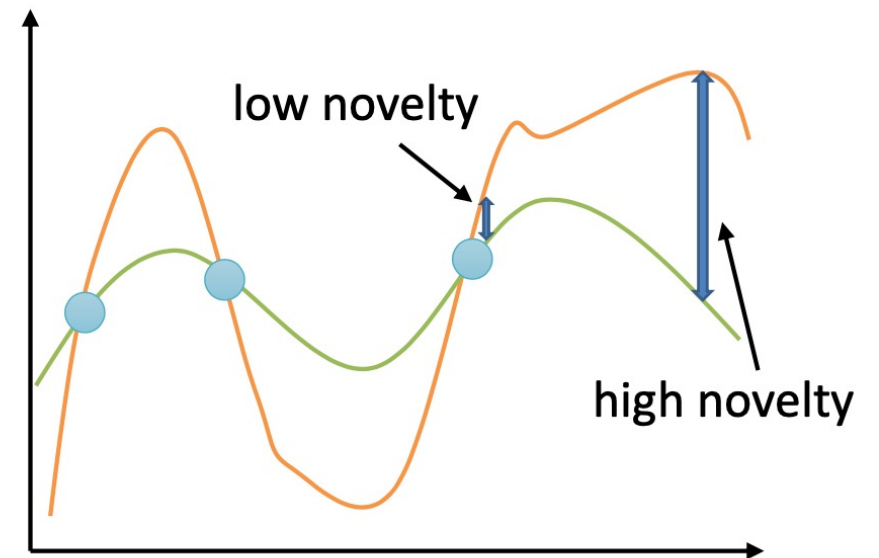
...and doesn't even need to output great densities

...just need to tell if state is **novel** or not!

let's say we have some **target** function  $f^*(\mathbf{s}, \mathbf{a})$

given our buffer  $\mathcal{D} = \{(\mathbf{s}_i, \mathbf{a}_i)\}$ , fit  $\hat{f}_\theta(\mathbf{s}, \mathbf{a})$

use  $\mathcal{E}(\mathbf{s}, \mathbf{a}) = \|\hat{f}_\theta(\mathbf{s}, \mathbf{a}) - f^*(\mathbf{s}, \mathbf{a})\|^2$  as bonus



# Heuristic estimation of counts via errors

what should we use for  $f^*(\mathbf{s}, \mathbf{a})$ ?

one common choice: set  $f^*(\mathbf{s}, \mathbf{a}) = \mathbf{s}'$  – i.e., next state prediction

even simpler:  $f^*(\mathbf{s}, \mathbf{a}) = f_\phi(\mathbf{s}, \mathbf{a})$ , where  $\phi$  is a *random* parameter vector



# Posterior sampling in deep RL

Thompson sampling:

$$\theta_1, \dots, \theta_n \sim \hat{p}(\theta_1, \dots, \theta_n)$$

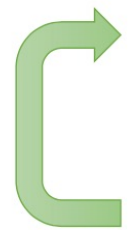
$$a = \arg \max_a E_{\theta_a}[r(a)]$$

What do we sample?

How do we represent the distribution?

bandit setting:  $\hat{p}(\theta_1, \dots, \theta_n)$  is distribution over *rewards*

MDP analog is the  $Q$ -function!



1. sample  $Q$ -function  $Q$  from  $p(Q)$
2. act according to  $Q$  for one episode
3. update  $p(Q)$

since  $Q$ -learning is off-policy, we don't care which  $Q$ -function was used to collect data

how can we represent a distribution over functions?

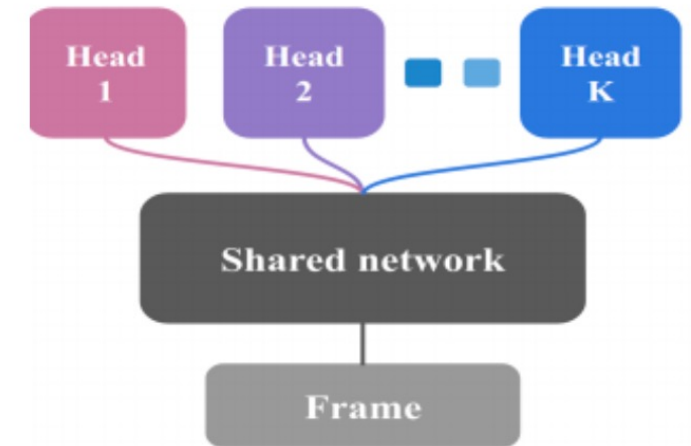
# Bootstrap

given a dataset  $\mathcal{D}$ , resample with replacement  $N$  times to get  $\mathcal{D}_1, \dots, \mathcal{D}_N$

train each model  $f_{\theta_i}$  on  $\mathcal{D}_i$

to sample from  $p(\theta)$ , sample  $i \in [1, \dots, N]$  and use  $f_{\theta_i}$

training  $N$  big neural nets is expensive, can we avoid it?



Osband et al. "Deep Exploration via Bootstrapped DQN"

# Bootstrap

---

**Algorithm 1** Bootstrapped DQN

---

```
1: Input: Value function networks  $Q$  with  $K$  outputs  $\{Q_k\}_{k=1}^K$ . Masking distribution  $M$ .  
2: Let  $B$  be a replay buffer storing experience for training.  
3: for each episode do  
4:   Obtain initial state from environment  $s_0$   
5:   Pick a value function to act using  $k \sim \text{Uniform}\{1, \dots, K\}$   
6:   for step  $t = 1, \dots$  until end of episode do  
7:     Pick an action according to  $a_t \in \arg \max_a Q_k(s_t, a)$   
8:     Receive state  $s_{t+1}$  and reward  $r_t$  from environment, having taking action  $a_t$   
9:     Sample bootstrap mask  $m_t \sim M$   
10:    Add  $(s_t, a_t, r_{t+1}, s_{t+1}, m_t)$  to replay buffer  $B$   
11:   end for  
12: end for
```

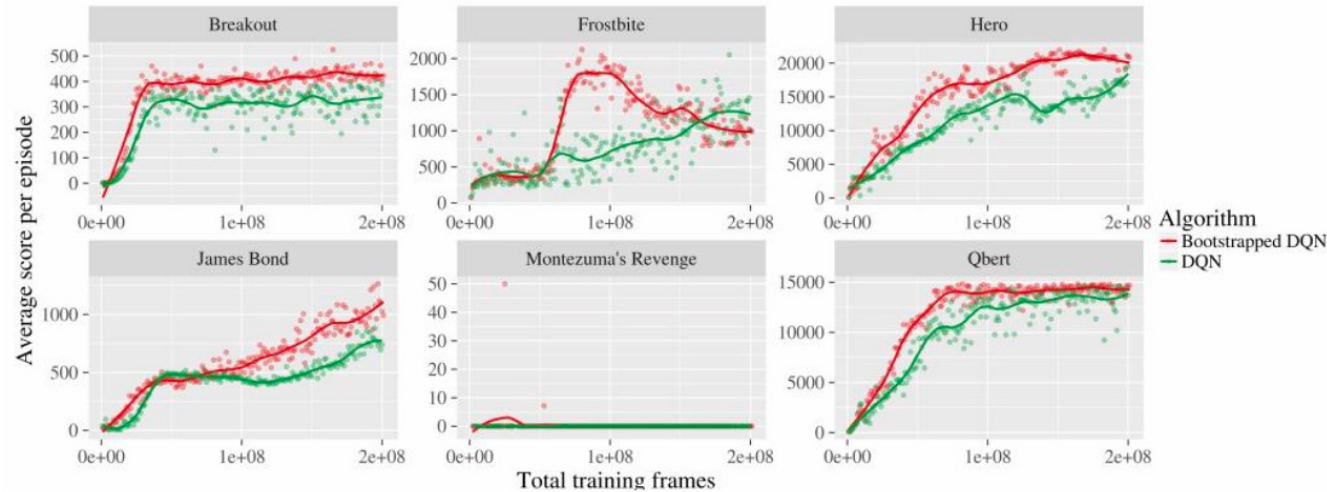
---

Osband et al. “Deep Exploration via Bootstrapped DQN”

# Why does this work?

Exploring with random actions (e.g., epsilon-greedy): oscillate back and forth, might not go to a coherent or interesting place

Exploring with random Q-functions: commit to a randomized but internally consistent strategy for an entire episode



+ no change to original reward function

- very good bonuses often do better

Osband et al. "Deep Exploration via Bootstrapped DQN"

# Go-Explore Method

## First return, then explore

Adrien Ecoffet<sup>\*1</sup>, Joost Huizinga<sup>\*1</sup>, Joel Lehman<sup>1</sup>, Kenneth O. Stanley<sup>1</sup> & Jeff Clune<sup>1,2</sup>

<sup>1</sup>*Uber AI, San Francisco, CA, USA*

<sup>2</sup>*OpenAI, San Francisco, CA, USA (work done at Uber AI)*

*\* These authors contributed equally to this work*

**Authors' note:** This is the pre-print version of this work. You most likely want the updated, published version, which can be found as an unformatted PDF at <https://adrien.ecoffet.com/files/go-explore-nature.pdf> or as a formatted web-only document at <https://tinyurl.com/Go-Explore-Nature>.

Please cite as:

Ecoffet, A., Huizinga, J., Lehman, J., Stanley, K.O. and Clune, J. First return, then explore. *Nature* **590**, 580–586 (2021). <https://doi.org/10.1038/s41586-020-03157-9>

IJ 16 Sep 2021

# Go-Explore Method

**Sparse Rewards:** Requires a **lot** of exploration;  
e.g. task: reaching coffee machine.  
define reward as **whether or not** the machine has been reached.

**Dense Rewards:**  
e.g. define reward as the inverse Euclidean distance to the machine.

- **Deceptive**
- **Reward Hacking**

# Solving two major issues





Solves two issue:

**Detachment:** the algorithm **loses track** of previously **seen** and **promising** states

**Derailment:** **exploratory** mechanisms of the policy prevents it from reaching **previously visited** states

# State Archive

- Builds an **archive** of the different states visited in the environment.
- **Probabilistically** selects a state to return to from the **archive**.
- Goes back (i.e. returns) to that state, **then explores** from that state.
- Updates the archive with **all novel states** encountered

State	Cell	Score	Visits	...	Prob:
		0	108	...	3%
		500	7	...	27%
...					...

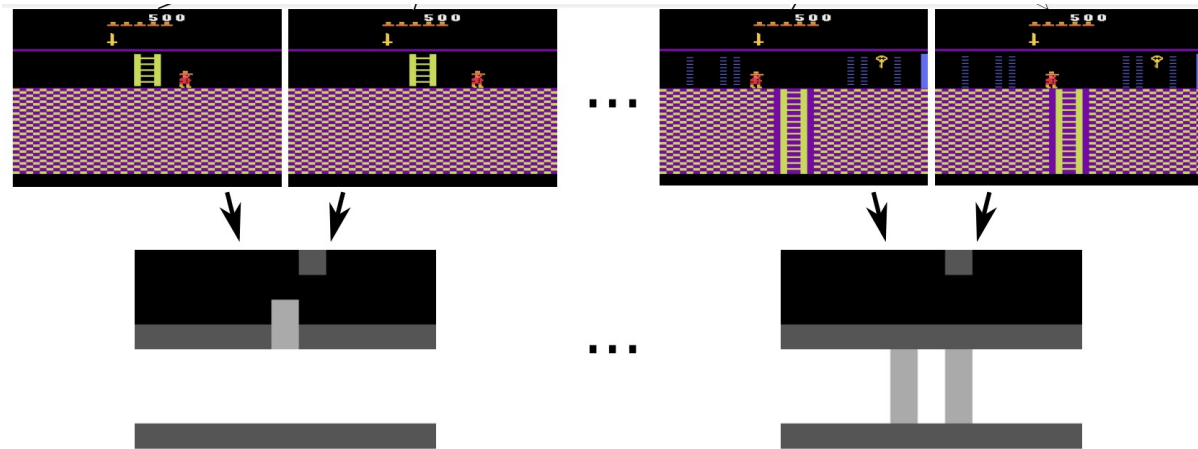


# How to build archive

Non-trivial environments have **too many** states to store explicitly.

Similar states are **grouped** into **cells**.

States are only considered **novel** if they are in a cell that does **not yet exist** in the archive.



To maximize performance, if a state maps to an already known cell, but is associated with a **better trajectory**, that state and its associated trajectory will **replace** the state and trajectory currently associated with that cell.

# How to return to a state?

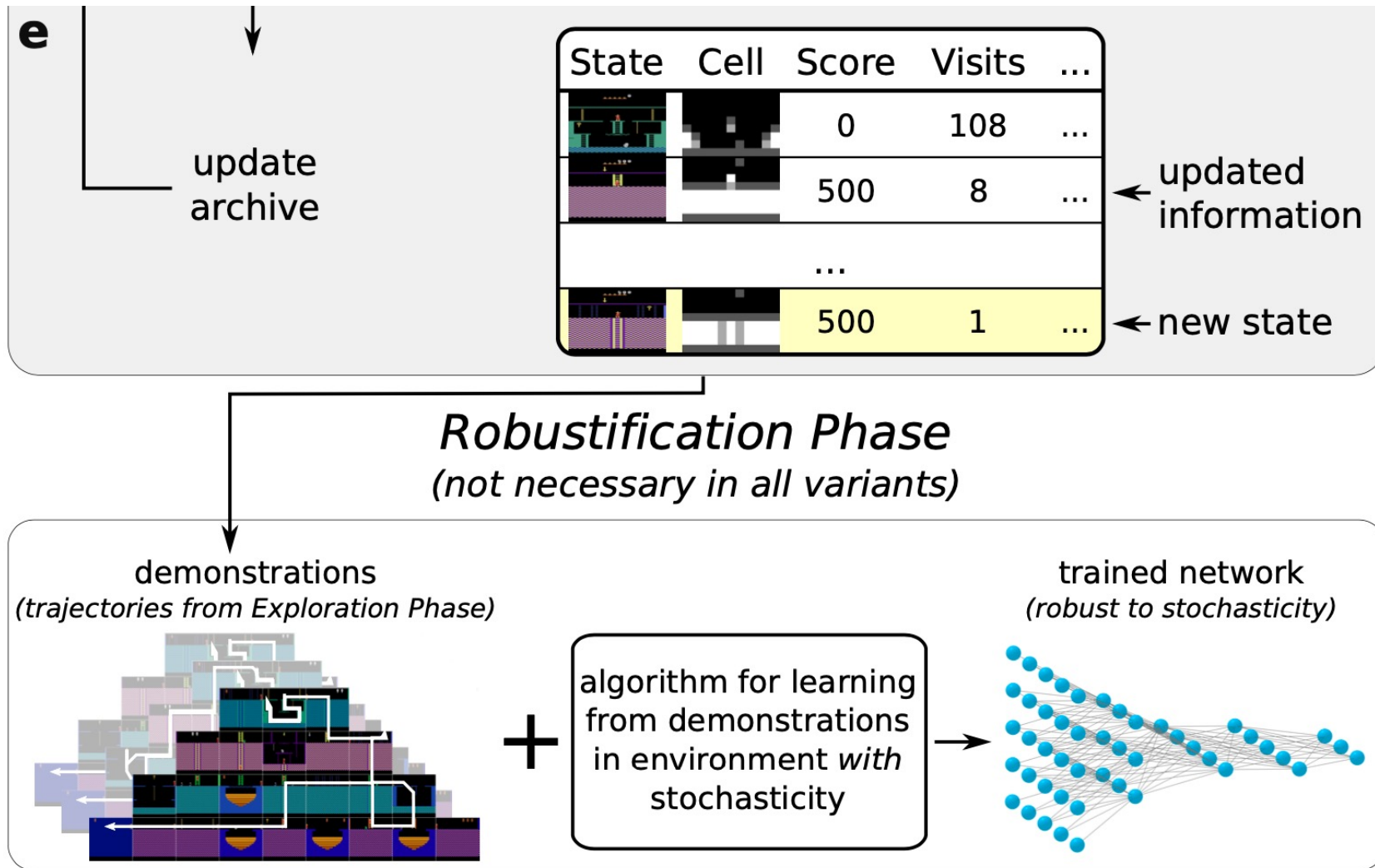
The **inner state** of any simulator can be **saved** and **restored** at a later time, making it possible to **instantly return** to a **previously seen state**.

Else, one can train a **goal-conditioned** policy for this purpose.

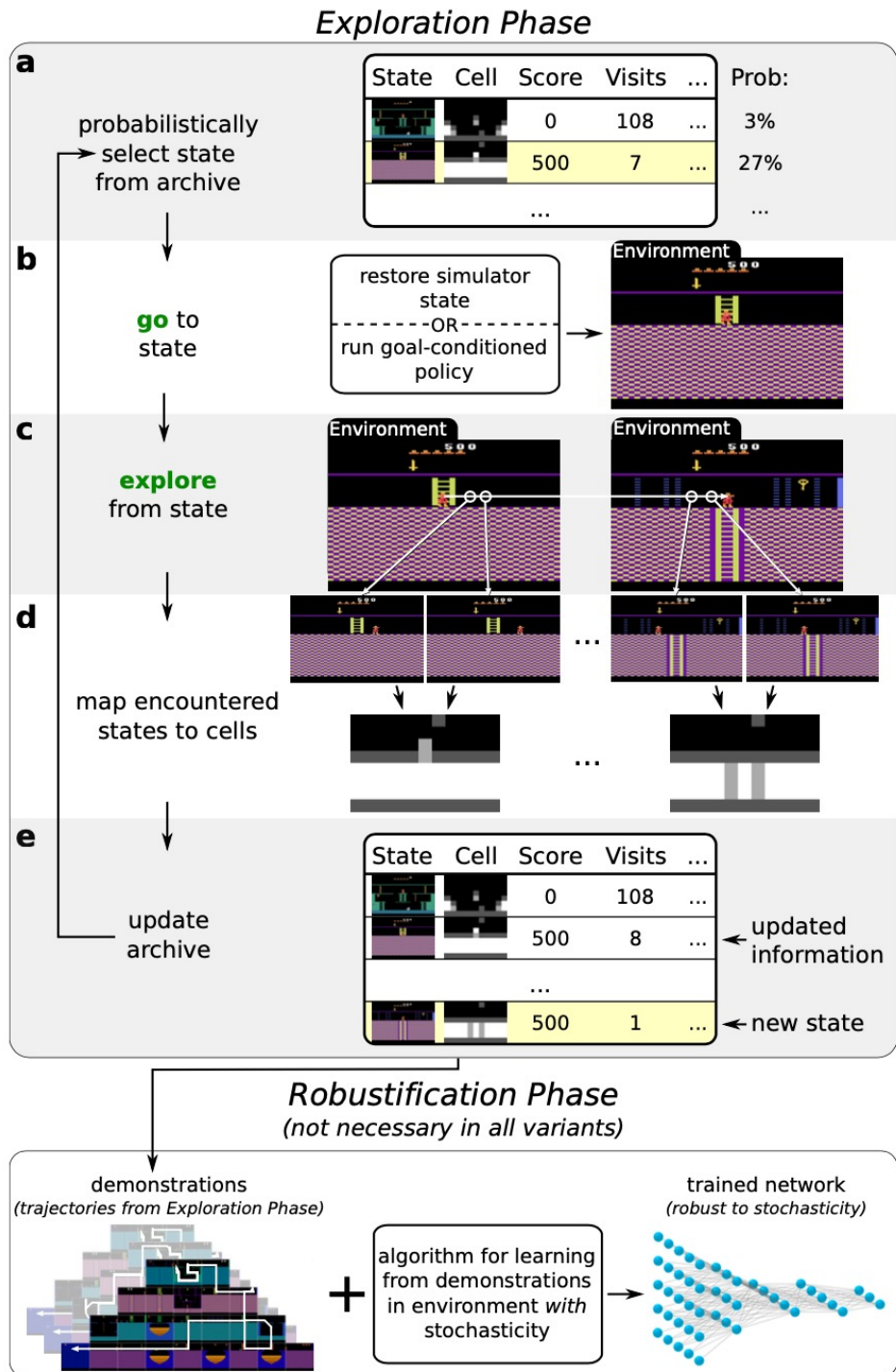
However, returning without a policy also means that this exploration process, which we call the exploration phase, does **not** produce a policy **robust** to the **inherent stochasticity** of the real world.

After the exploration phase is complete, these trajectories make it possible to train a **robust** and **high-performing** policy by **Learning from Demonstrations (LfD)**.

# How to return to a state?



# All in one view



# Results

Game	Exploration Phase	Robustification Phase	SOTA	Avg. Human
Berzerk	131,216	<b>197,376</b>	1,383	2,630
Bowling	247	<b>260</b>	69	160
Centipede	613,815	<b>1,422,628</b>	10,166	12,017
Freeway	34	<b>34</b>	<b>34</b>	30
Gravitar	13,385	<b>7,588</b>	3,906	3,351
MontezumaRevenge	24,758	<b>43,791</b>	11,618	4,753
Pitfall	6,945	<b>6,954</b>	0	6,463
PrivateEye	60,529	<b>95,756</b>	26,364	69,571
Skiing	-4,242	<b>-3,660</b>	-10,386	-4,336
Solaris	20,306	<b>19,671</b>	3,282	12,326
Venture	3,074	<b>2,281</b>	1,916	1,187

# Results

