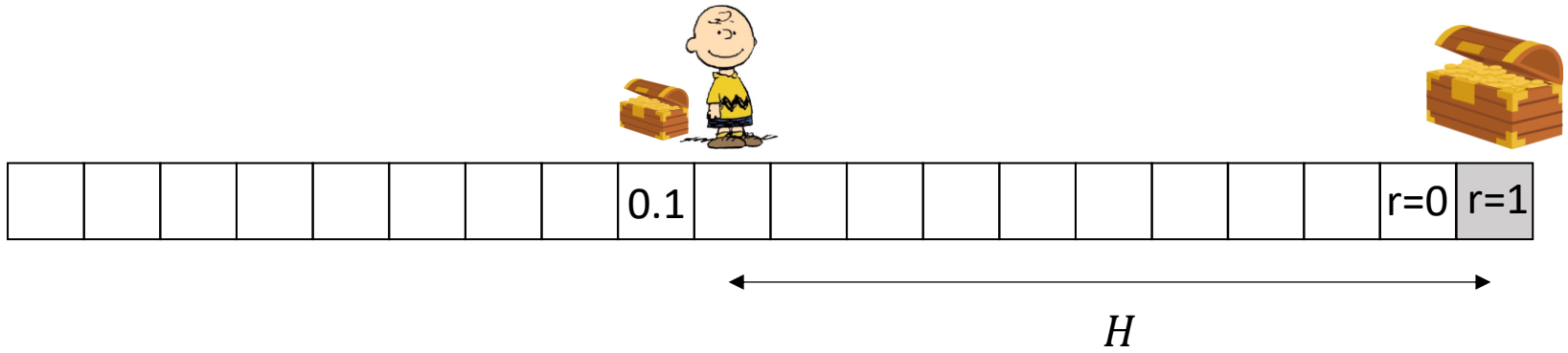
Exploration in MDPs

Chen-Yu Wei

State-Space Exploration in MDPs



Environment:

- Fixed-horizon MDP with episode length H
- Initial state at 0
- A single rewarding state at state H
- Actions: Go LEFT or RIGHT

Suppose we perform DQN with ϵ-greedy with random initialization
⇒ On average, we need 2^H episodes to see the reward
(before that, we won't make any meaningful update)

Regret Analysis for MDPs?

- We have done regret analysis for several bandit algorithms:
  - Regression oracle + (ϵ-greedy or inverse gap weighting)
  - UCB
  - EXP3

- We did not really establish regret bounds for MDPs. We only argued:
  - Approximate value iteration: under the assumption that the data in replay buffer is exploratory
  - Approximate policy iteration: monotonically improvement

Regret Analysis for MDPs?

$$\mathbb{E}_{s\sim\rho}\left[V^{\pi^\star}(s)\right] - \mathbb{E}_{s\sim\rho}[V^{\pi}(s)]$$

$$= \sum_{s,a} d_\rho^{\pi}(s,a)\left(V^\star(s) - Q^\star(s,a)\right)$$

For VI-based algorithm (approximating $Q^\star$)

Approximating $Q^\star(s,a)$ requires the replay buffer to cover **wide range of** state-actions.

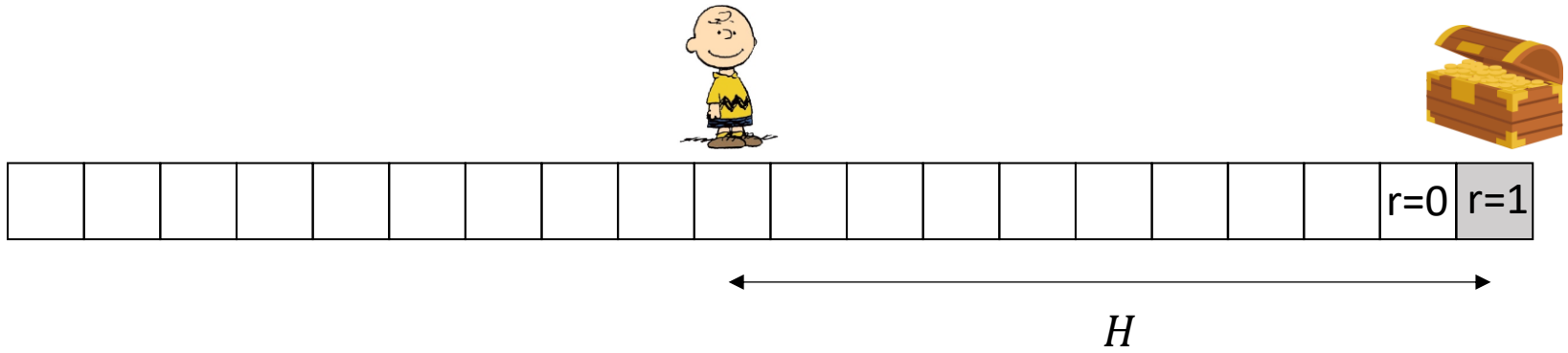
$$= \sum_{s,a} d_\rho^{\pi^\star}(s,a)\left(Q^{\pi}(s,a) - V^{\pi}(s)\right)$$

For PI-based algorithm (approximating Q^π)

Approximating $Q^\pi(s,a)$ only requires state-actions generated from current policy

But…

Regret Analysis for MDPs?



$$\sum_{s,a} d_\rho^\pi(s,a)\,(V^\star(s) - Q^\star(s,a))$$

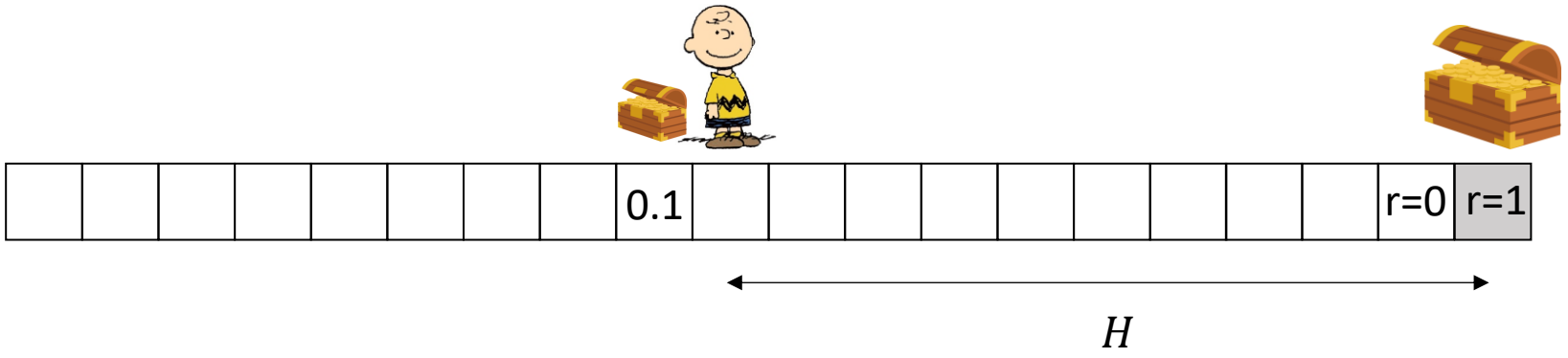
$$\sum_{s,a} d_\rho^{\pi^\star}(s,a)\,(Q^\pi(s,a) - V^\pi(s))$$

PI-based algorithm only tries to make $\sum_{s,a} d_\rho^{\pi_k}(s,a)\,(Q^\pi(s,a) - V^\pi(s))$ small.

It can only quickly find optimal policy when $d_\rho^{\pi_k} \approx d_\rho^{\pi^\star}$

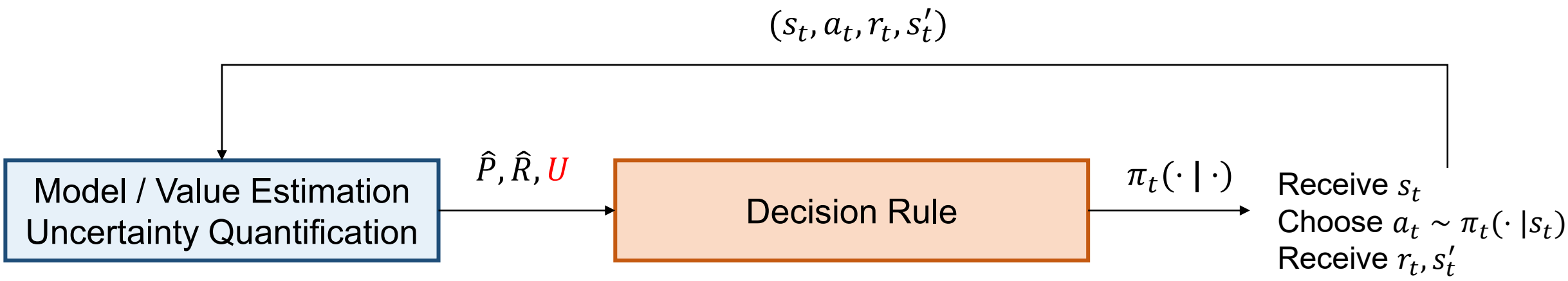
Insufficiency of algorithms we have discussed for MDPs

- Lack of **exploration over the state space** (we need **deep exploration**)
- This issue is particularly critical if
  - Local reward does not provide any information
  - Local reward provide misleading information



- Solution
  - Try to make the data (i.e., state-action) distribution close to $d^{\pi^\star}$
  - Try to visit as many states as possible (by quantifying the leaner's **uncertainty** about a state)

Exploration via Uncertainty Quantification



Exploration Bonus for Bandits (Optimism Principle)

- We have discussed this idea for action exploration – UCB.

Upper Confidence Bound

$$a_t = \text{argmax}_a \quad \hat{R}_t(a) + \sqrt{\frac{2\log(2/\delta)}{N_t(a)}}$$

$\hat{R}_t(a)$ = the empirical mean of arm a up to time $t-1$.
$N_t(a)$ = the number of times we draw arm a up to time $t-1$.

$$a_t = \text{argmax}_a \quad \hat{R}_t(a) + \sqrt{\frac{2\log(2/\delta)}{N_t(a)}}$$

Exploration Bonus for MDPs

UCB Value Iteration (UCBVI)

For episode $1, 2, \ldots, T$:

$\tilde{Q}_{H+1}(s,a) = 0 \quad \forall s, a$

For step $H, H-1, \ldots, 1$:

$$\tilde{Q}_h(s,a) \triangleq \hat{R}(s,a) + \sum_{s'} \hat{P}(s'|s,a) \max_{a'} \tilde{Q}_{h+1}(s',a') + H\sqrt{\frac{2\log(2/\delta)}{N_t(s,a)}} \qquad \forall s, a$$

Receive $s_1 \sim \rho$

For step $1, 2, \ldots, H$:

Take action $a_h = \text{argmax}_a \, \tilde{Q}_h(s_h, a)$

Receive $r_h = R(s_h, a_h) + \text{noise}, \quad s_{h+1} \sim P(\cdot\,|s_h, a_h)$

Exploration Bonus for MDPs

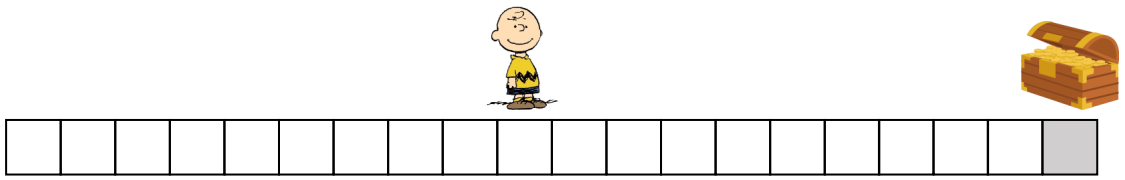
$$\tilde{Q}_h(s,a) \triangleq \hat{R}(s,a) + \sum_{s'} \hat{P}(s'|s,a) \max_{a'} \tilde{Q}_{h+1}(s',a') + H\sqrt{\frac{2\log(2/\delta)}{N_t(s,a)}} \quad \forall s,a$$

Exploration Bonus for MDPs

Theorem. Regret Bound of UCBVI

UCBVI ensures with high probability,

$$\text{Regret} = \sum_{t=1}^{T} \left(V^{\star}(s_{t,1}) - V^{\pi_t}(s_{t,1}) \right) \lesssim H\sqrt{SAT}.$$



Improving the required number of episodes from 2^H to $\text{poly}(H)$

Jaksch, Ortner, Auer. Near-Optimal Regret Bounds for Reinforcement Learning. 2010.

Azar, Osband, Munos. Minimax Regret Bounds for Reinforcement Learning. 2017.

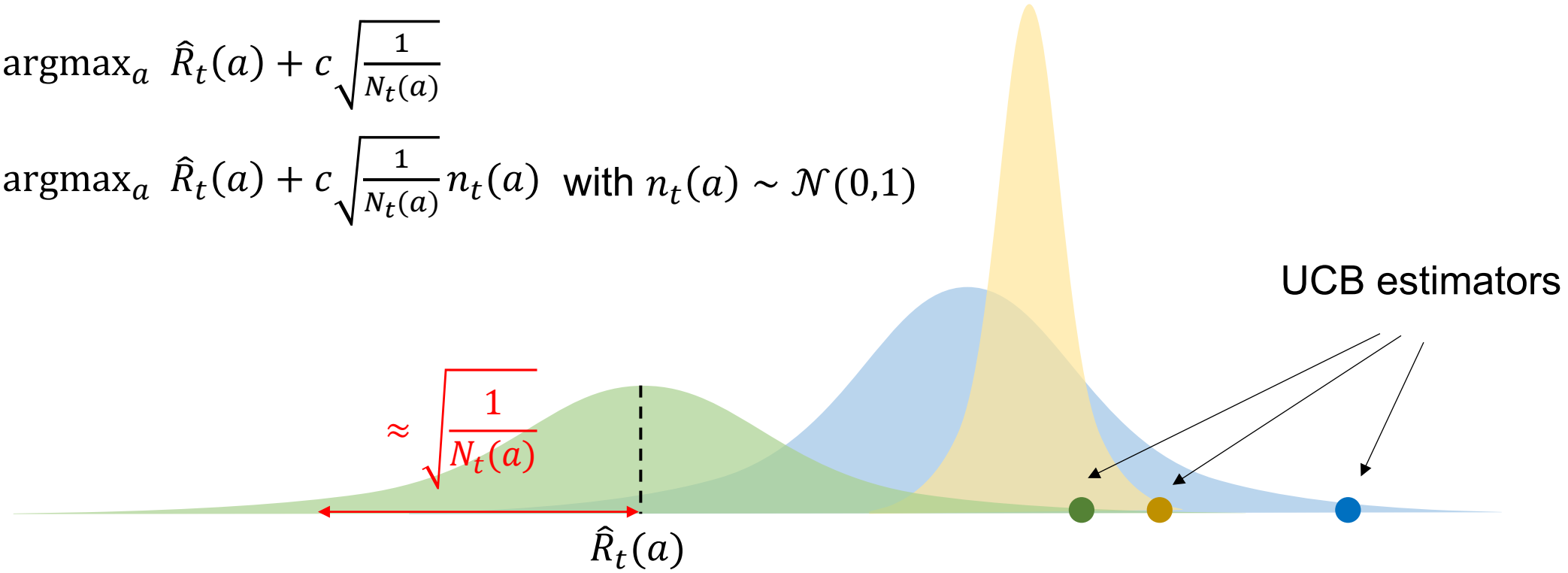
Thompson Sampling
(Posterior Sampling)

Bayesian interpretation:
Assume the reward mean $(\theta(1), \ldots, \theta(A))$ is drawn from a Gaussian distribution (prior distribution).
Then the **posterior distribution** is

$$P(\theta(a)|\mathcal{H}_t) = \mathcal{N}\left(\hat{R}_t(a), \frac{1}{N_t(a)}\right)$$

UCB: $a_t \approx \text{argmax}_a \ \hat{R}_t(a) + c\sqrt{\frac{1}{N_t(a)}}$

TS: $a_t \approx \text{argmax}_a \ \hat{R}_t(a) + c\sqrt{\frac{1}{N_t(a)}} n_t(a)$ with $n_t(a) \sim \mathcal{N}(0,1)$



Randomized Exploration for MDPs

Randomized Value Iteration

For episode $1, 2, \ldots, T$:

$\tilde{Q}_{H+1}(s,a) = 0 \quad \forall s, a$

For step $H, H-1, \ldots, 1$:

$$\tilde{Q}_h(s,a) \triangleq \hat{R}(s,a) + \sum_{s'} \hat{P}(s'|s,a) \max_{a'} \tilde{Q}_{h+1}(s',a') + H \sqrt{\frac{2\log(2/\delta)}{N_t(s,a)}} \underbrace{n_t(s,a)}_{\sim \mathcal{N}(0,1)}$$

Receive $s_1 \sim \rho$

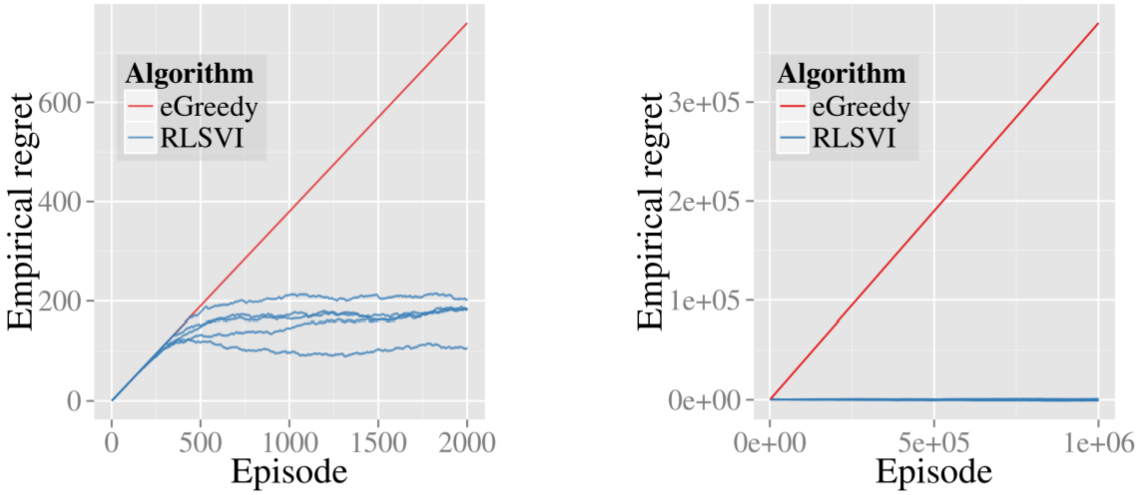
For step $1, 2, \ldots, H$:

Take action $a_h = \text{argmax}_a \, \tilde{Q}_h(s_h, a)$

Receive $r_h = R(s_h, a_h) + \text{noise}, \quad s_{h+1} \sim P(\cdot\,|s_h, a_h)$

Osband, Van Roy, Wen. Generalization and Exploration via Randomized Value Functions. 2014.

Randomized Exploration for MDPs



(a) First 2000 episodes (b) First 10^6 episodes

Figure 2. Efficient exploration on a 50-chain

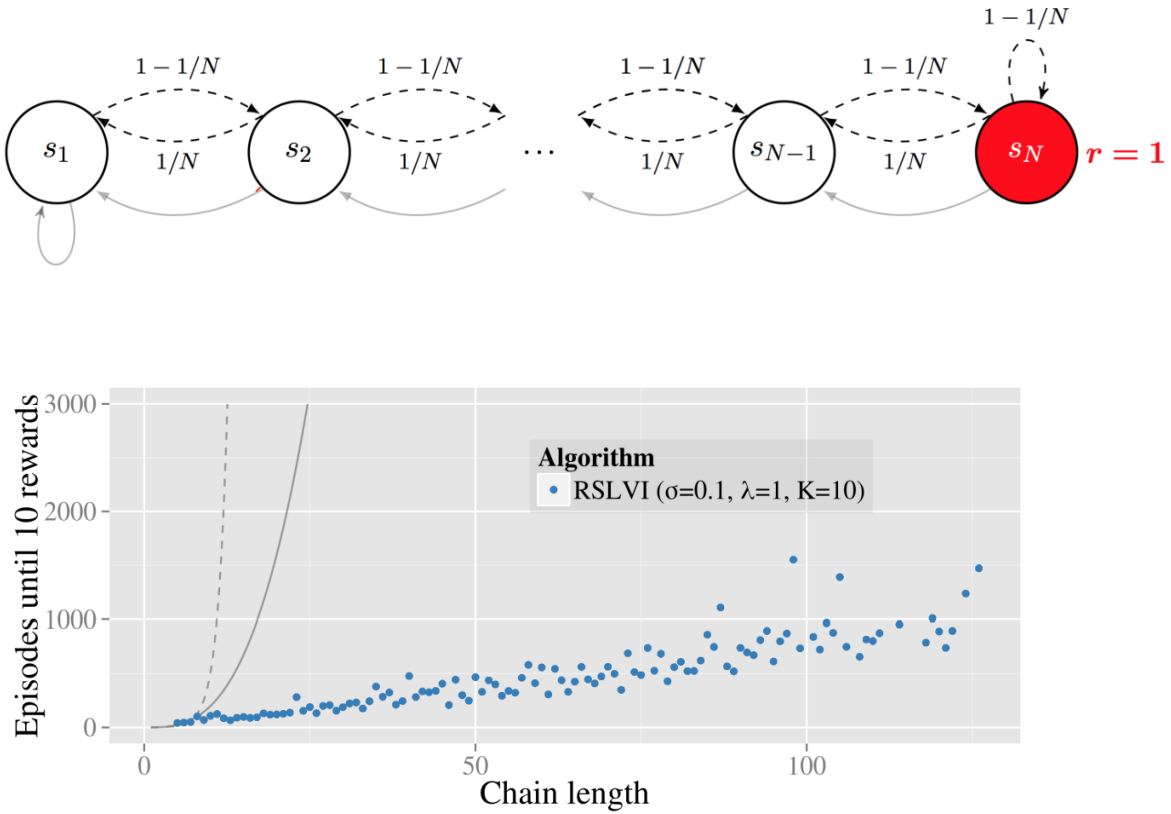


Figure 3. RLSVI learning time against chain length.

Osband, Van Roy, Wen. Generalization and Exploration via Randomized Value Functions. 2014.

Common Approaches of Exploration

- Optimistic Exploration
  - Upper Confidence Bound
- Randomized Exploration
  - Thompson Sampling (Posterior Sampling)
- Information-Directed Exploration

Exploration in Large State Spaces with Function Approximation

UCB / TS with Given State-Action Features

Suppose for any (s, a), we have access to a feature vector $\phi(s, a) \in \mathbb{R}^d$.

Then instead of counting the #visits to every state-action, we can evaluate the **novelty of the feature**.

$$\Lambda_t = \sum_{i<t} \sum_{h=1}^{H} \phi(s_{ih}, a_{ih}) \phi(s_{ih}, a_{ih})^\top$$

LSVI-UCB $$\tilde{Q}_h(s,a) \triangleq \hat{R}(s,a) + \sum_{s'} \hat{P}(s'|s,a) \max_{a'} \tilde{Q}_{h+1}(s',a') + c \cdot \sqrt{\phi(s,a)\Lambda_t^{-1}\phi(s,a)}$$

Jin et al. Provably efficient reinforcement learning with linear function approximation. 2019.

RLSVI $$\tilde{Q}_h(s,a) \triangleq \hat{R}(s,a) + \sum_{s'} \hat{P}(s'|s,a) \max_{a'} \tilde{Q}_{h+1}(s',a') + c \cdot \mathcal{N}(0, \phi(s,a)\Lambda_t^{-1}\phi(s,a))$$

Zanette et al. Frequentist Regret Bounds for Randomized Least-Squares Value Iteration. 2019.

How to Adapt These Ideas to General Cases?

Ideas from UCB:

or $\frac{1}{N(s,a)}$ (theoretically better for deterministic environment)

1. $\tilde{R}(s,a) = \hat{R}(s,a) + \frac{1}{\sqrt{N(s,a)}}$ where $N(s,a) \approx$ Amount of prior visit to (s,a)
2. $\tilde{R}(s,a) = \hat{R}(s,a) + e(s,a)$ where $e(s,a) \approx$ Prediction error on $\hat{R}(s,a)$ and $\hat{P}(\cdot\,|s,a)$

Ideas from TS:

3. $\tilde{R}(s,a) = \hat{R}(s,a) +$ noise whose variance scales with the uncertainty of $\hat{R}(s,a)$ and $\hat{P}(\cdot\,|s,a)$

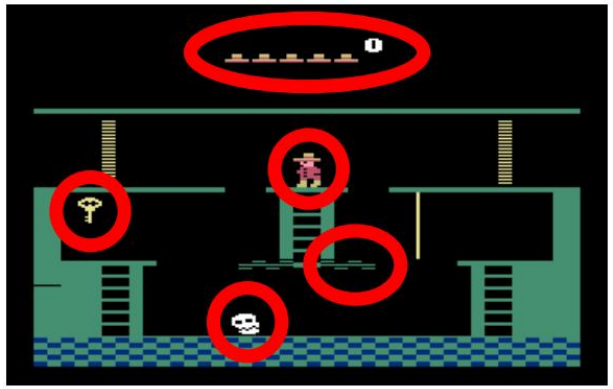
Ideas from Information-directed Sampling:

4. $\tilde{R}(s,a) = \hat{R}(s,a) + \lambda \underbrace{\mathrm{KL}\big(\mathcal{P}(\cdot\,|\mathcal{H}_t, s, a, s'), \mathcal{P}(\cdot\,|\mathcal{H}_t)\big)}_{\text{Information gain}}$

After all these, just perform standard RL algorithm over $\tilde{R}$.

1. Bonus from the Number of Prior Visits

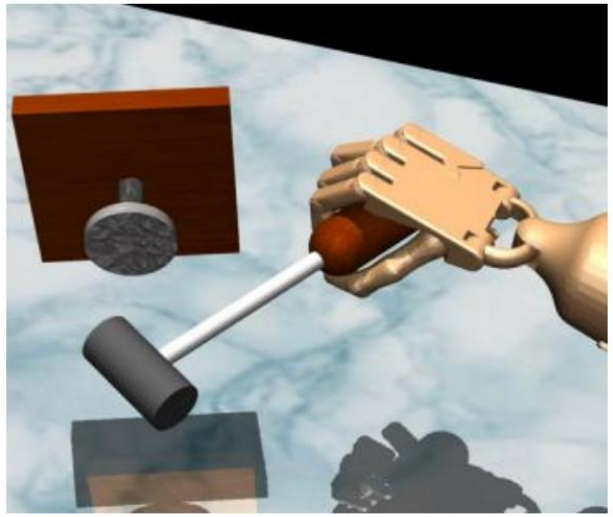
Pseudo Count



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:

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

count

probability/density

total states visited

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

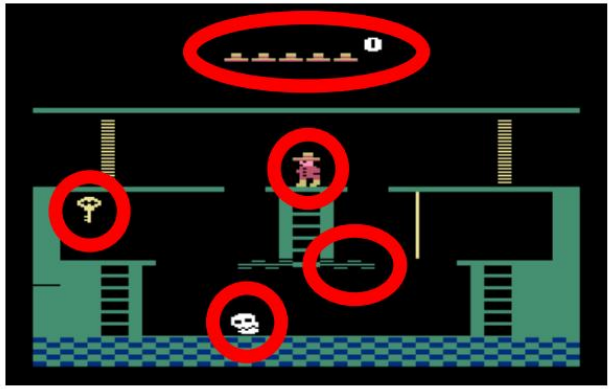
$$P'(\mathbf{s}) = \frac{N(\mathbf{s}) + 1}{n + 1}$$

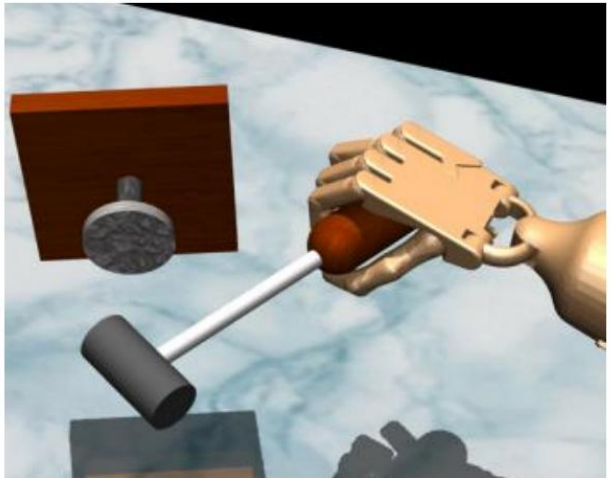
can we get $p_\theta(\mathbf{s})$ and $p_{\theta'}(\mathbf{s})$ to obey these equations?

Bellemare et al. Unifying Count-Based Exploration and Intrinsic Motivation. 2016

(Source: Sergey Levine's Deep RL)

Pseudo Count





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

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

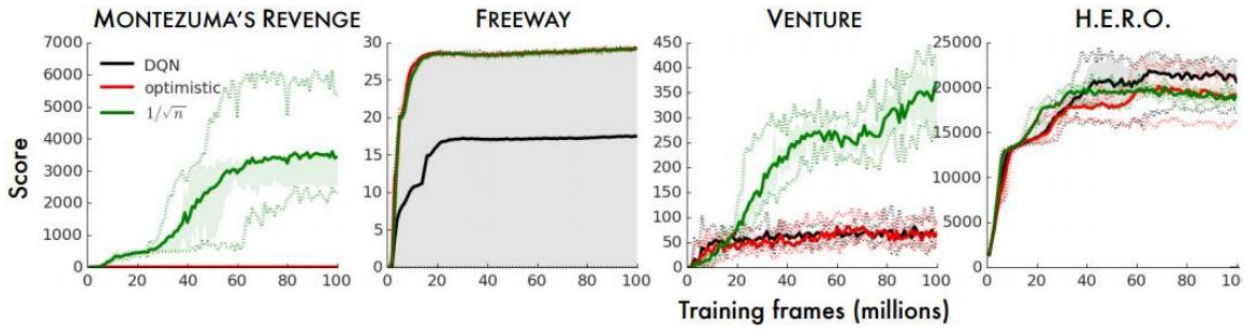
two equations and two unknowns!

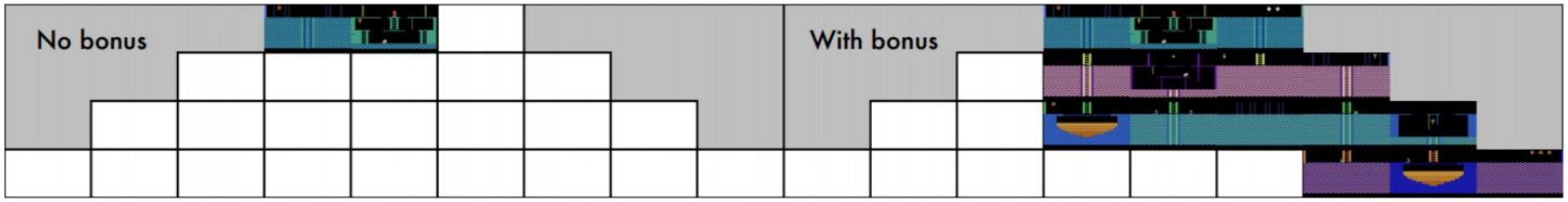
$$\hat{N}(\mathbf{s}_i) = \hat{n} p_\theta(\mathbf{s}_i) \qquad \hat{n} = \frac{1 - p_{\theta'}(\mathbf{s}_i)}{p_{\theta'}(\mathbf{s}_i) - p_\theta(\mathbf{s}_i)} p_\theta(\mathbf{s}_i)$$

Bellemare et al. Unifying Count-Based Exploration and Intrinsic Motivation. 2016

(Source: Sergey Levine's Deep RL)

Pseudo Count





Bellemare et al. Unifying Count-Based Exploration and Intrinsic Motivation. 2016

(Source: Sergey Levine's Deep RL)

Hash

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

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

shorter codes = more hash collisions

similar states get the same hash? maybe

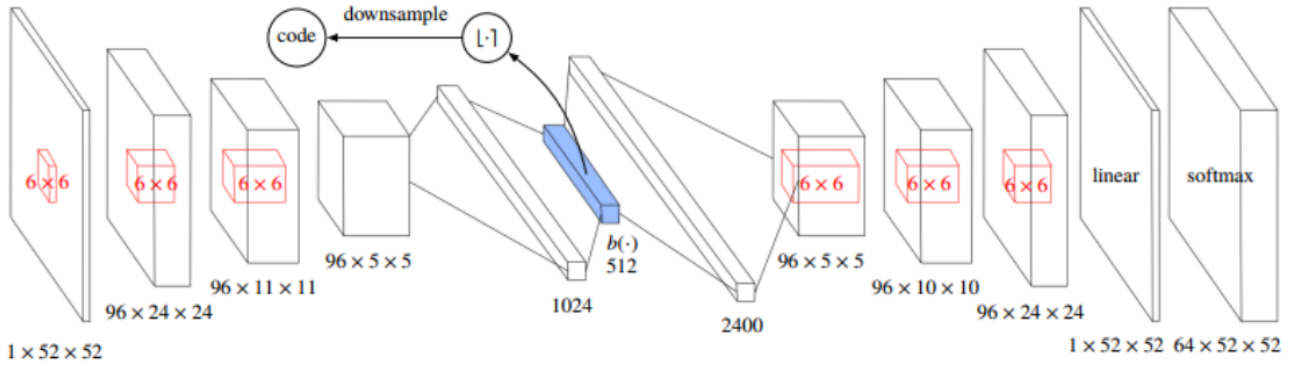
$$\phi(s) = \text{sgn}(Ag(s)) \in \{-1, 1\}^k, \tag{2}$$

where $g : \mathcal{S} \to \mathbb{R}^D$ is an optional preprocessing function and A is a $k \times D$ matrix with i.i.d. entries drawn from a standard Gaussian distribution $\mathcal{N}(0, 1)$. The value for k controls the granularity: higher values lead to fewer collisions and are thus more likely to distinguish states.

#Exploration: A Study of Count-Based Exploration for Deep Reinforcement Learning. 2017.

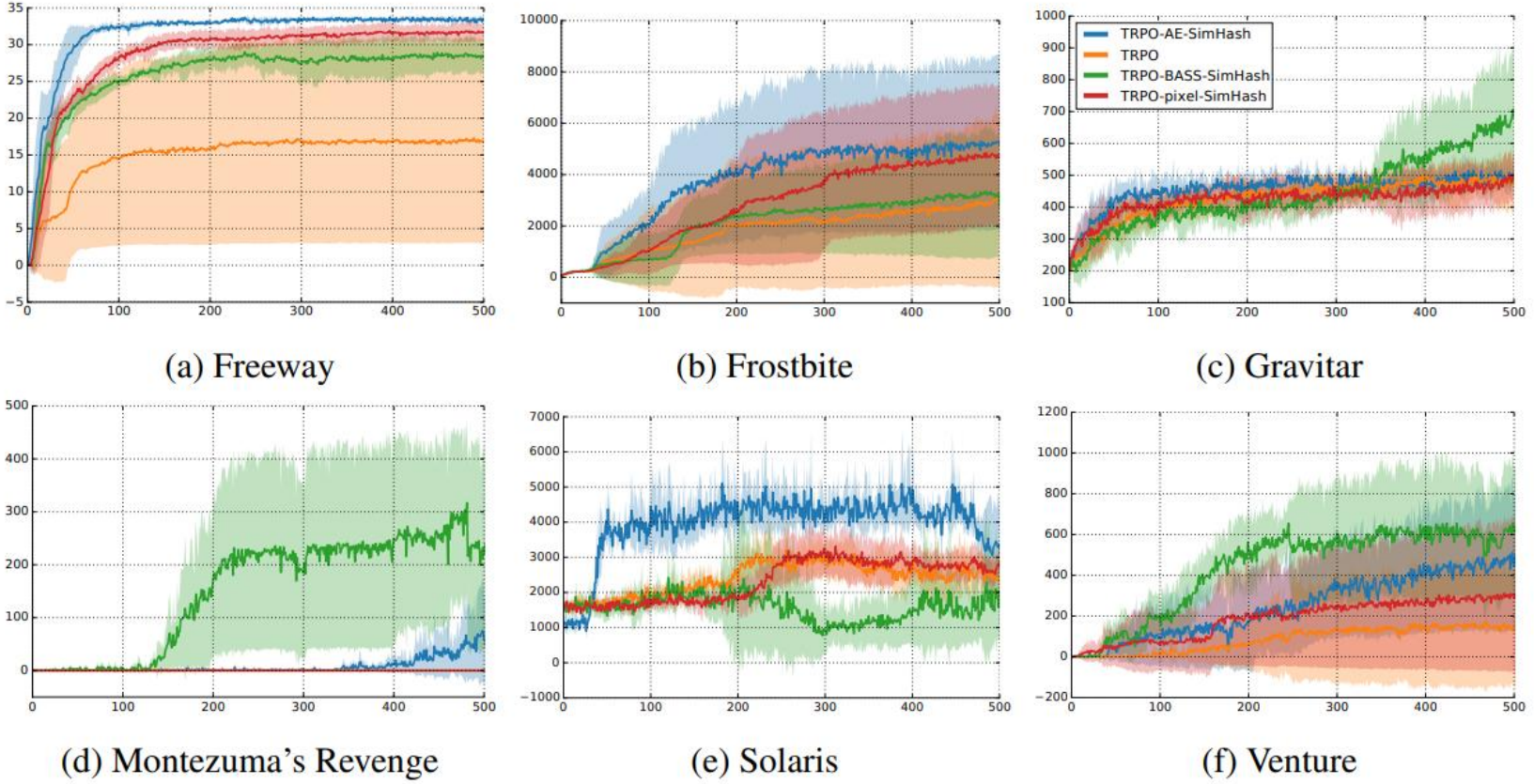
Hash

improve the odds by *learning* a compression:



#Exploration: A Study of Count-Based Exploration for Deep Reinforcement Learning. 2017.

Hash



(a) Freeway (b) Frostbite (c) Gravitar

(d) Montezuma's Revenge (e) Solaris (f) Venture

#Exploration: A Study of Count-Based Exploration for Deep Reinforcement Learning. 2017.

2. Bonus from Prediction Error

Bonus from Prediction Error

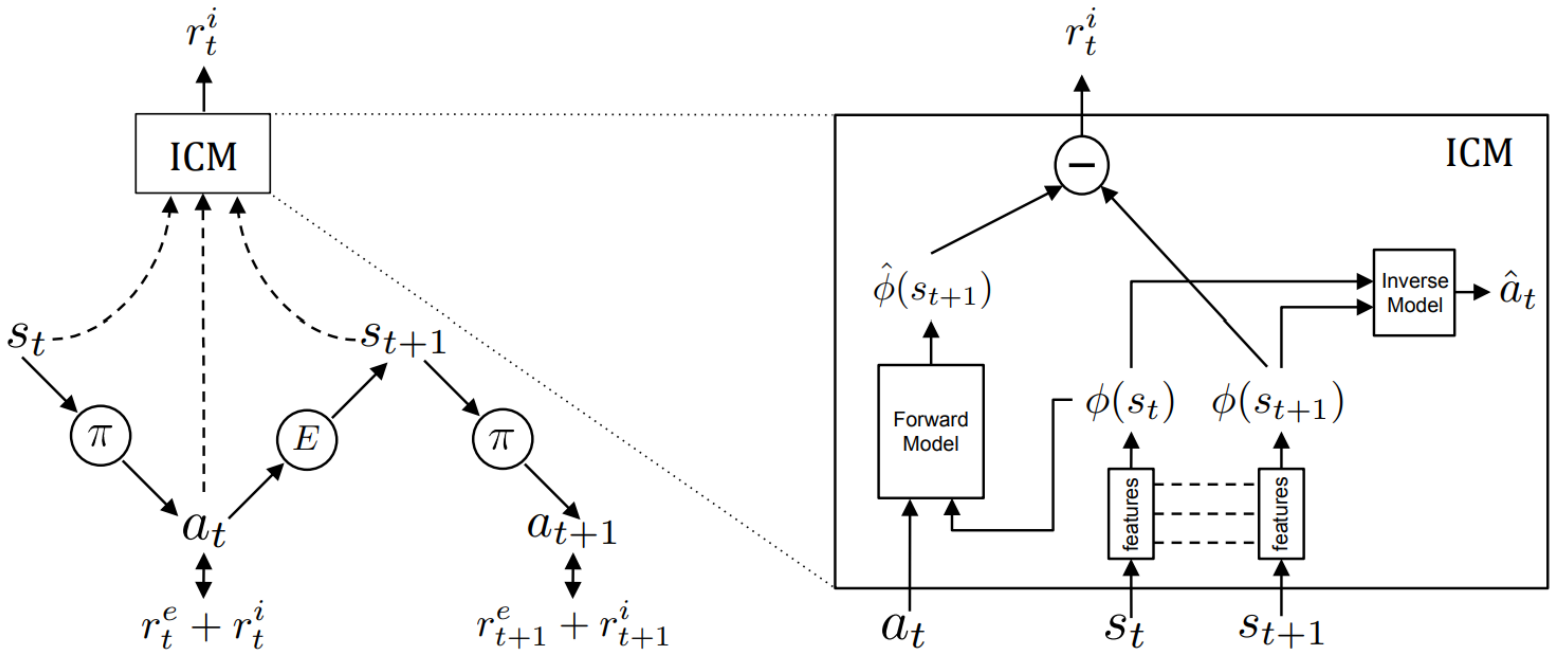
Ideally, we would like to estimate $\left\|\hat{P}(\cdot\,|s,a) - P(\cdot\,|s,a)\right\|$ and set it as bonus.

However, we don't know the ground-truth transition, so the best we can do is try to **predict the next state**.

There are some issues if we naively do this:

1. For stochastic environments where transitions are random, we will never have small prediction error for next state.
2. For many environments, some part of the state is uncontrollable by the learner (e.g., movement of the clouds in the background).

Intrinsic Curiosity Module



Task 1: Given s_t and s_{t+1}, predict a_t: make the feature ϕ capture action-related dynamics

Task 2: Given $\phi(s_t)$ and a_t, predict $\phi(s_{t+1})$

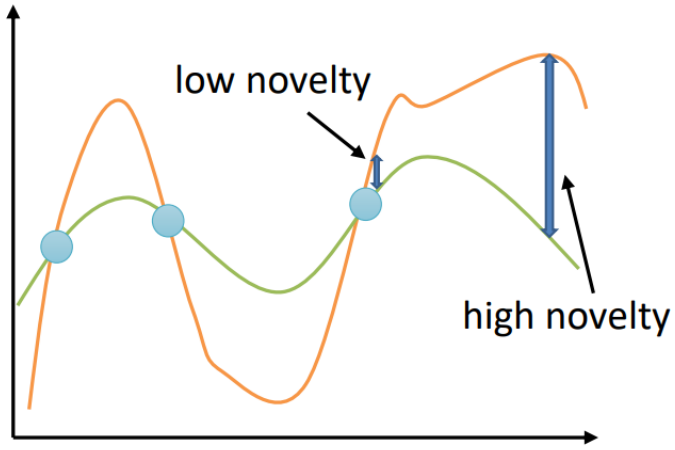
Pathak et al. Curiosity-driven Exploration by Self-supervised Prediction. 2017.

Random Network Distillation

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

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

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



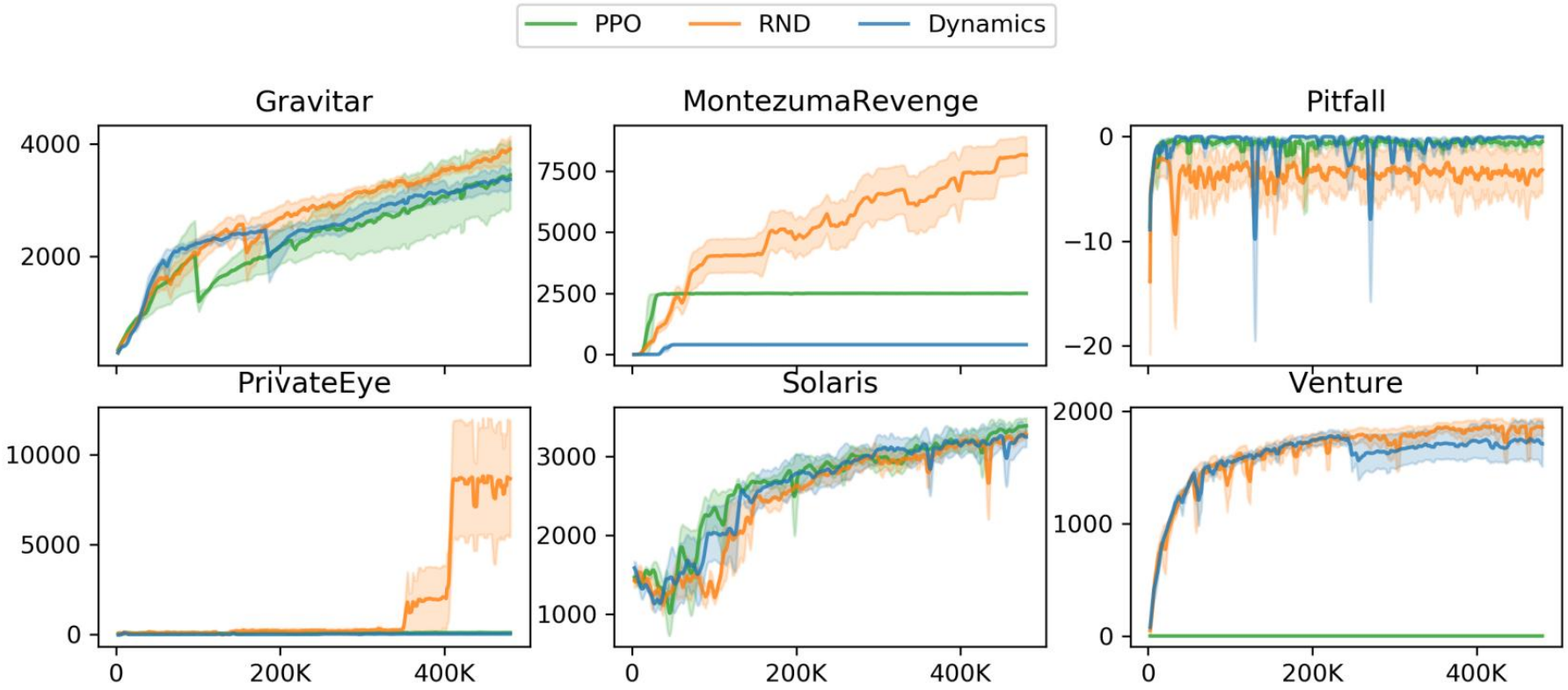
what should we use for $f^\star(\mathbf{s}, \mathbf{a})$?

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

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

Burda et al. Exploration by Random Network Distillation. 2018.

Random Network Distillation



Burda et al. Exploration by Random Network Distillation. 2018.

3. Thompson Sampling

Recall: Randomized Value Iteration

Randomized Value Iteration

For episode $1, 2, \ldots, T$:

$\tilde{Q}_{H+1}(s,a) = 0 \quad \forall s, a$

For step $H, H-1, \ldots, 1$:

$$\tilde{Q}_h(s,a) \triangleq \hat{R}(s,a) + \sum_{s'} \hat{P}(s'|s,a) \max_{a'} \tilde{Q}_{h+1}(s',a') + H \sqrt{\frac{2\log(2/\delta)}{N_t(s,a)}} \underbrace{n_t(s,a)}_{\sim \mathcal{N}(0,1)}$$

Receive $s_1 \sim \rho$

For step $1, 2, \ldots, H$:

Take action $a_h = \text{argmax}_a \, \tilde{Q}_h(s_h, a)$

Receive $r_h = R(s_h, a_h) + \text{noise}, \quad s_{h+1} \sim P(\cdot\,|s_h, a_h)$

Osband, Van Roy, Wen. Generalization and Exploration via Randomized Value Functions. 2014.

Recall: Randomized Value Iteration

$$\tilde{Q}_h(s,a) \triangleq \hat{R}(s,a) + \sum_{s'} \hat{P}(s'|s,a) \max_{a'} \tilde{Q}_{h+1}(s',a') + n_t(s,a)$$

Adapting this idea to DQN:

$$\theta = \operatorname*{argmin}_{\theta} \sum_{(s,a,r,s') \in \mathcal{B}} \left(r + \max_{a'} Q_{\bar{\theta}}(s',a') + n_t(s,a) - Q_\theta(s,a) \right)^2 \qquad (*)$$

Notice that different noise gives different θ.

Direct generalization from Randomized VI (not easy to implement)

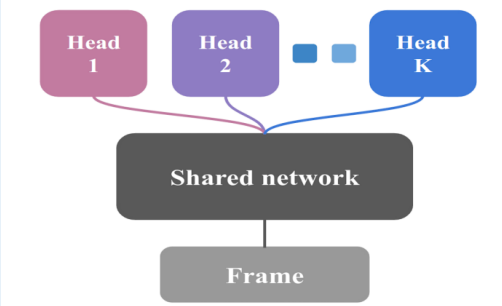
Θ = Space of θ's

In each episode, sample a $\theta \in \Theta$ with the distribution following (*), and execute $\pi(s) = \operatorname*{argmax}_a Q_\theta(s,a)$

Bootstrapped DQN

Osband et al. Deep Exploration via Bootstrapped DQN. 2016.
Osband et al. Randomized Prior Functions for Deep Reinforcement Learning. 2018.

Randomly initialize K instances of DQN $\theta_1, \dots, \theta_K$
(each θ_i has their own target network $\bar{\theta}_i$ and replay buffer $\mathcal{B}_i$).



(a) Shared network architecture

For each episode:

Randomly sample $i \sim \text{Unif}\{1,2,\dots,K\}$

Execute $\pi(s) = \max_a Q_{\theta_i}(s,a)$ in the whole episode.

Randomly place the obtained (s,a,r,s') in some/all replay buffers.

Update all DQN parameters.

Bootstrapped DQN

Osband et al. Deep Exploration via Bootstrapped DQN. 2016.
Osband et al. Randomized Prior Functions for Deep Reinforcement Learning. 2018.

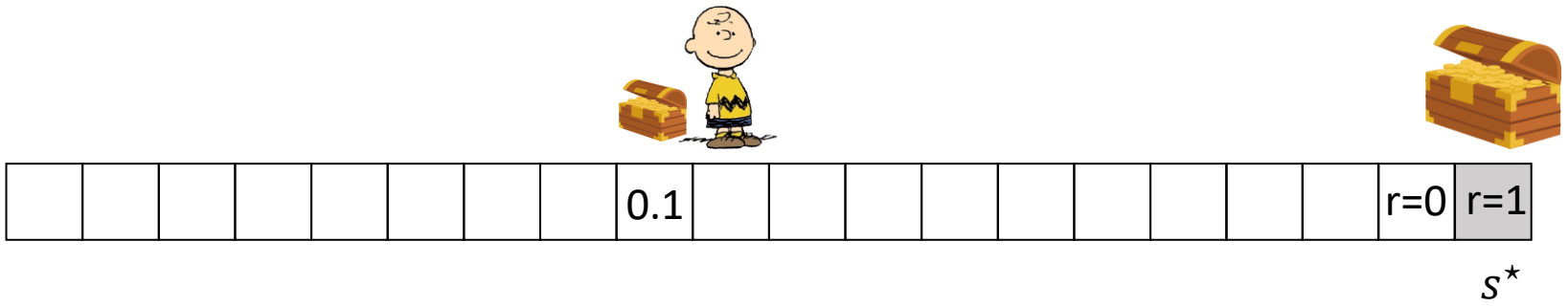
Some intuitions:

- The random initialization makes $Q_{\theta_1}(s,a), \ldots, Q_{\theta_K}(s,a)$ all very different.
  We can view them as associated with different initial noise $n_1(s,a)$.
- Over the course of training, for (s,a)'s that are more often visited, their effective magnitude of $n_t(s,a)$ decreases (because we train those DQNs without adding more noise).
- For (s,a)'s that are not often visited, their effective magnitude of $n_t(s,a)$ remains high.
- **Why does this perform deep exploration?** For a particular state s, if $\max_a Q_{\theta_i}(s,a)$ is initialized high but has not been visited many times before, the training of θ_i will propagate this high value to other state and encourage the learner to reach s from other states.

Bootstrapped DQN

Osband et al. Deep Exploration via Bootstrapped DQN. 2016.
Osband et al. Randomized Prior Functions for Deep Reinforcement Learning. 2018.

- In the toy example, as long as **one of the K DQNs** initializes $s^\star$ (or some states close to it) with a high value, then it can help the learner explore to $s^\star$.
- In this example, roughly we need $K = \mathrm{O}(\text{number of states})$ to achieve this effect.



Bootstrapped DQN

Osband et al. Deep Exploration via Bootstrapped DQN. 2016.
Osband et al. Randomized Prior Functions for Deep Reinforcement Learning. 2018.

"Deep Sea" Exploration

- **Stylized "chain" domain testing "deep exploration":**
  - *State = N x N grid, observations 1-hot.*
  - *Start in top left cell, fall one row each step.*
  - *Actions {0,1} map to left/right in each cell.*
  - ***"left" has reward = 0, "right" has reward = -0.1/N***
  - ***... but if you make it to bottom right you get +1.***
- Only one policy (out of more than 2^N) positive return.
- **ε-greedy / Boltzmann / policy gradient / are useless.**