

# Exploration (Part 1)

CS 285: Deep Reinforcement Learning, Decision Making, and Control

Sergey Levine

# Class Notes

1. Homework 4 due Wednesday!

# Today's Lecture

1. What is exploration? Why is it a problem?
  2. Multi-armed bandits and theoretically grounded exploration
  3. Optimism-based exploration
  4. Posterior matching exploration
  5. Information-theoretic exploration
- Goals:
    - Understand what the exploration is
    - Understand how theoretically grounded exploration methods can be derived
    - Understand how we can do exploration in deep RL in practice

# What's the problem?

this is easy (mostly)



this is impossible



## Why?

# 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!

# Put yourself in the algorithm's shoes



## Mao

- “the only rule you may be told is this one”
  - Incur a penalty when you break a rule
  - Can only discover rules through trial and error
  - Rules don't always make sense to you
- 
- Temporally extended tasks like Montezuma's revenge become increasingly difficult based on
    - How extended the task is
    - How little you know about the rules
  - Imagine if your goal in life was to win 50 games of Mao...
  - (and you didn't know this in advance)

# Another example



Learned Policies

# 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?
- Actually the same problem:
  - Exploitation: doing what you *know* will yield highest reward
  - Exploration: doing things you haven't done before, in the hopes of getting even higher reward



# Exploration and exploitation examples

- Restaurant selection
  - **Exploitation**: go to your favorite restaurant
  - **Exploration**: try a new restaurant
- Online ad placement
  - **Exploitation**: show the most successful advertisement
  - **Exploration**: show a different random advertisement
- Oil drilling
  - **Exploitation**: drill at the best known location
  - **Exploration**: drill at a new location

# Exploration is hard

Can we derive an **optimal** exploration strategy?

what does optimal even mean?

regret vs. Bayes-optimal strategy? more on this later...

multi-armed bandits  
(1-step stateless  
RL problems)

contextual bandits  
(1-step RL problems)

small, finite MDPs  
(e.g., tractable planning,  
model-based RL setting)

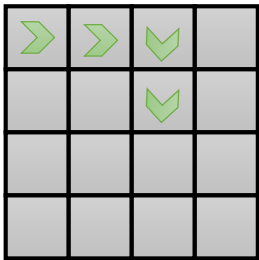
large, infinite MDPs,  
continuous spaces

←  
theoretically tractable

→  
theoretically intractable

# What makes an exploration problem tractable?

multi-arm bandits  
contextual bandits



small, finite MDPs



large or infinite MDPs

can formalize exploration  
as POMDP identification  
policy learning is trivial  
even with POMDP

can frame as Bayesian model  
identification, reason explicitly  
about value of information

optimal methods don't work  
...but can take inspiration from  
optimal methods in smaller settings  
use hacks

# Bandits

What's a bandit anyway?



the drosophila of exploration problems



$$\mathcal{A} = \{\text{pull arm}\}$$

$$r(\text{pull arm}) = ?$$



$$\mathcal{A} = \{\text{pull}_1, \text{pull}_2, \dots, \text{pull}_n\}$$

$$r(a_n) = ?$$

$$\text{assume } r(a_n) \sim \underline{p(r|a_n)}$$

unknown *per-action* reward distribution!

# Let's play!

CAN YOU BEAT THE BANDIT ALGORITHMS?

**CONFIGURATION**

Drugs:  Patients:  **PLAY**

**PLOT**

**MEDICAL TESTING**

**TRIAL RESULTS**

Trial complete, please see plot below or start again.

Timestep: 39. Drug: 4. Patient: Die

Timestep: 38. Drug: 4. Patient: Die

Timestep: 37. Drug: 4. Patient: Die

Timestep: 36. Drug: 4. Patient: Die

- Drug prescription problem
- Bandit arm = drug (1 of 4)
- Reward
  - 1 if patient lives
  - 0 if patient dies
  - (stakes are high)
- How well can you do?

<http://iosband.github.io/2015/07/28/Beat-the-bandit.html>

# How can we define the bandit?

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

- 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

how do we measure goodness of exploration algorithm?

regret: difference from optimal policy at time step  $T$ : 
$$\text{Reg}(T) = T E[r(a^*)] - \sum_{t=1}^T r(a_t)$$

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

actual reward of action  
actually taken

# How can we beat the bandit?

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

expected reward of best action  
(the best we can hope for in expectation)      ↗

↖      actual reward of action  
actually taken

- Variety of relatively simple strategies
- Often can provide theoretical guarantees on regret
  - Variety of optimal algorithms (up to a constant factor)
  - But empirical performance may vary...
- Exploration strategies for more complex MDP domains will be inspired by these 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):

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

number of times we  
picked this action

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



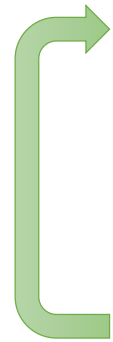
# Probability matching/posterior sampling

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”

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

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

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

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

# Why should we care?

- Bandits are easier to analyze and understand
- Can derive foundations for exploration methods
- Then apply these methods to more complex MDPs
- Not covered here:
  - Contextual bandits (bandits with state, essentially 1-step MDPs)
  - Optimal exploration in small MDPs
  - Bayesian model-based reinforcement learning (similar to information gain)
  - Probably approximately correct (PAC) exploration

Break

# Classes of exploration methods in deep RL

- Optimistic exploration:
  - new state = good state
  - requires estimating state visitation frequencies or novelty
  - typically realized by means of exploration bonuses
- Thompson sampling style algorithms:
  - learn distribution over Q-functions or policies
  - sample and act according to sample
- Information gain style algorithms
  - reason about information gain from visiting new states

# Optimistic exploration in RL

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

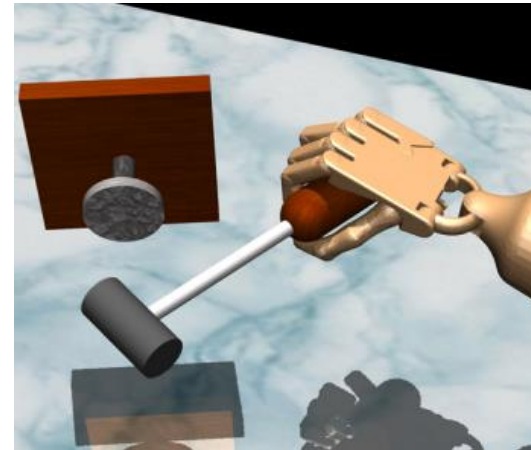
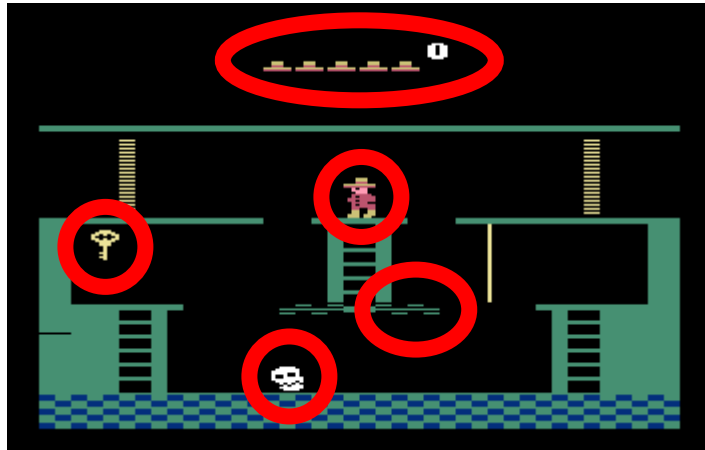
- need to tune bonus weight



# The trouble with counts

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

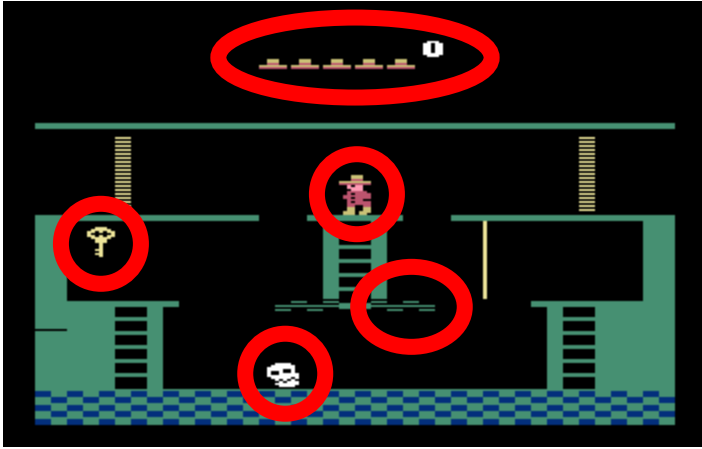
But wait... what's a count?



Uh oh... we never see the same thing twice!

But some states are more similar than others

# Fitting generative models

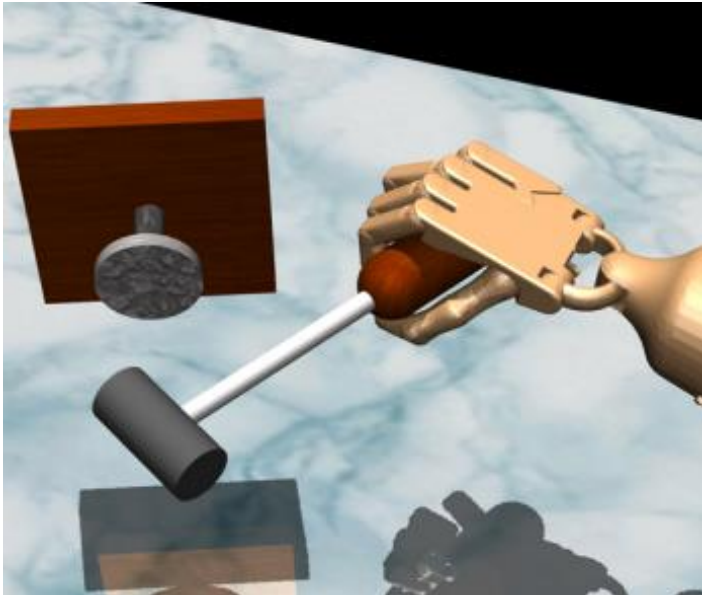


idea: fit a density model  $p_{\theta}(\mathbf{s})$  (or  $p_{\theta}(\mathbf{s}, \mathbf{a})$ )

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

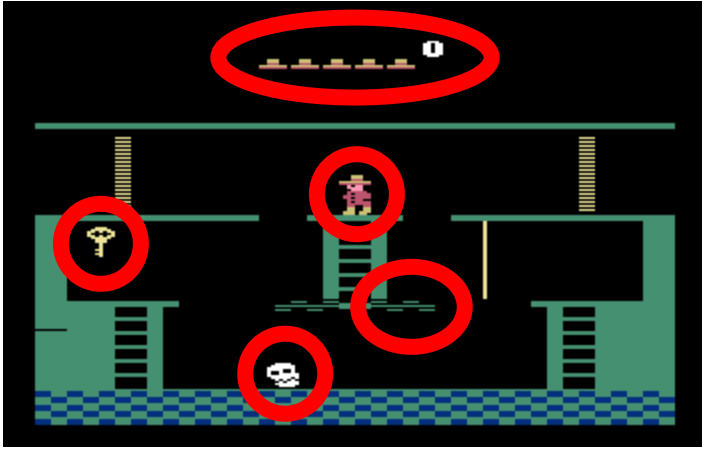
$$P(\mathbf{s}) = \frac{N(\mathbf{s})}{n}$$

probability/density  $\nwarrow$   $\nearrow$  count  $\nwarrow$  total states visited

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

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

# Exploring with pseudo-counts



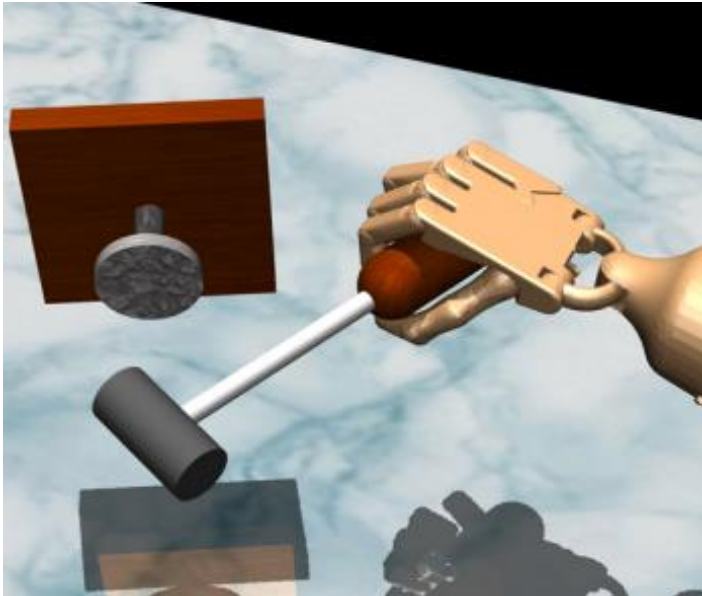
fit model  $p_{\theta}(\mathbf{s})$  to all states  $\mathcal{D}$  seen so far

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

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

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

two equations and two unknowns!

$$\hat{N}(\mathbf{s}_i) = \hat{n} p_{\theta}(\mathbf{s}_i)$$

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

# What kind of bonus to use?

Lots of functions in the literature, inspired by optimal methods for bandits or small MDPs

UCB:

$$\mathcal{B}(N(\mathbf{s})) = \sqrt{\frac{2 \ln n}{N(\mathbf{s})}}$$

MBIE-EB (Strehl & Littman, 2008):

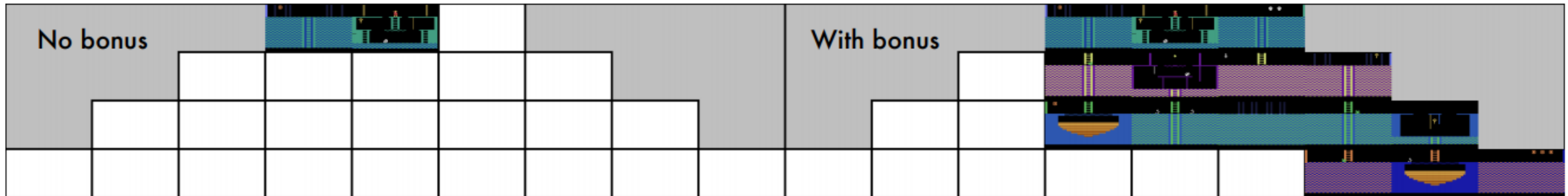
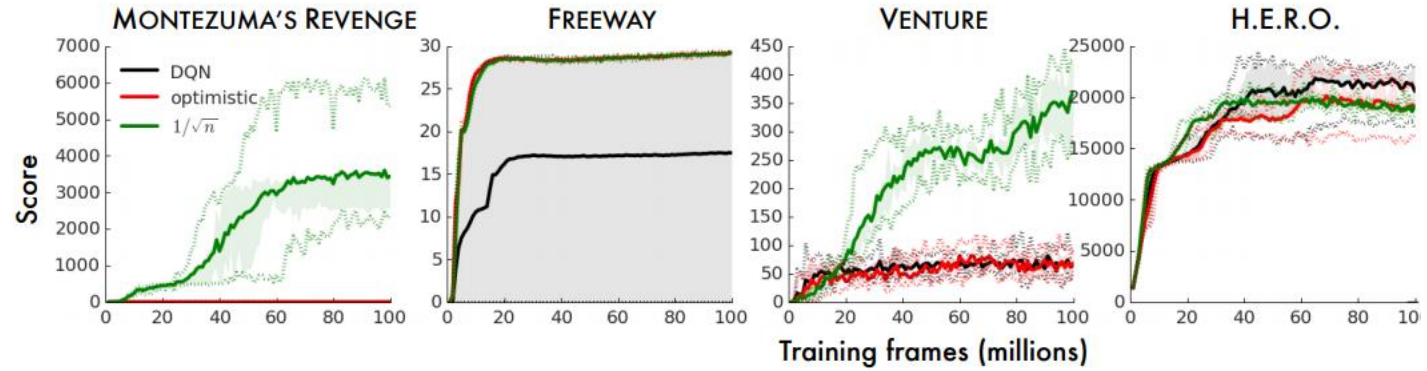
$$\mathcal{B}(N(\mathbf{s})) = \sqrt{\frac{1}{N(\mathbf{s})}}$$

BEB (Kolter & Ng, 2009):

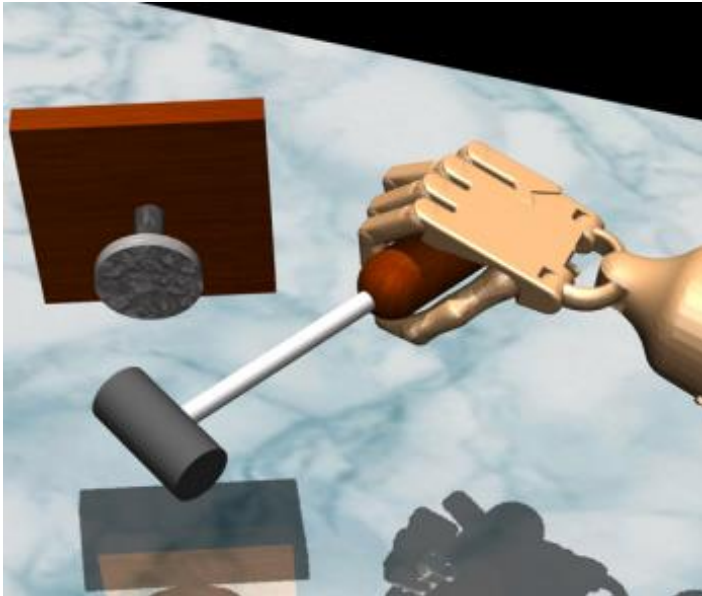
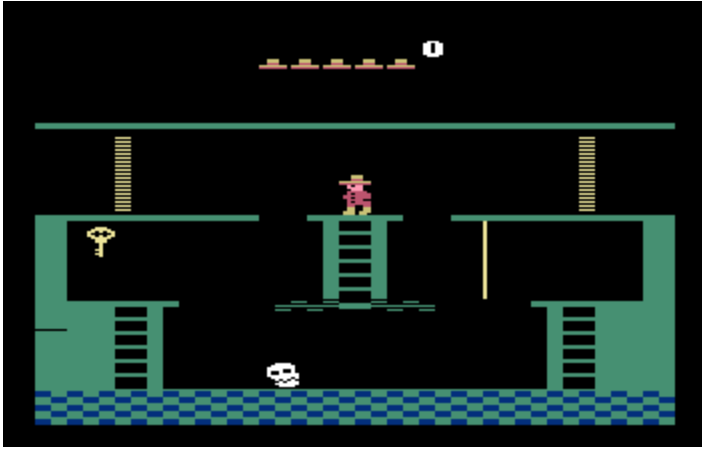
$$\mathcal{B}(N(\mathbf{s})) = \frac{1}{N(\mathbf{s})}$$

← this is the one used by Bellemare et al. '16

# Does it work?



# What kind of model to use?



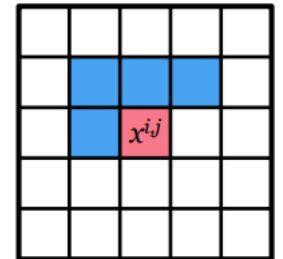
$$p_{\theta}(\mathbf{s})$$

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



# Counting with hashes

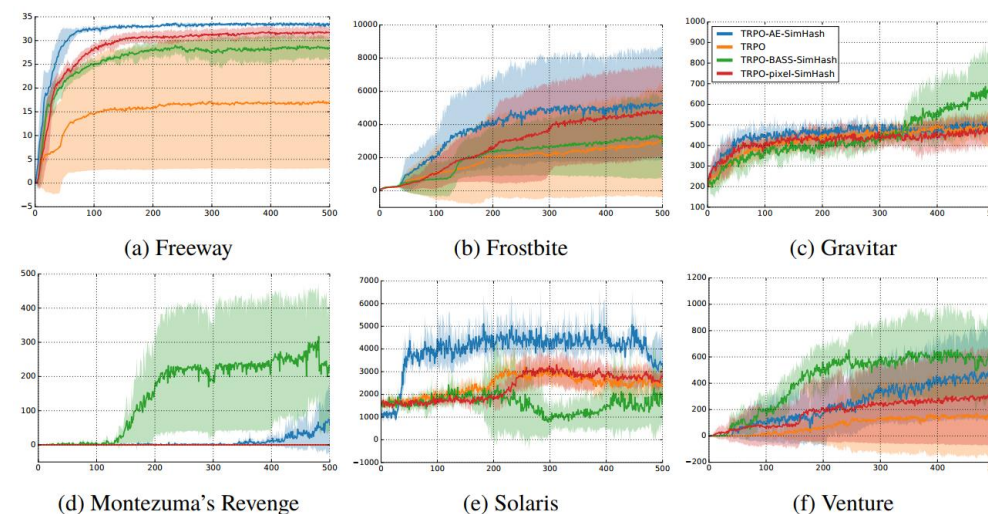
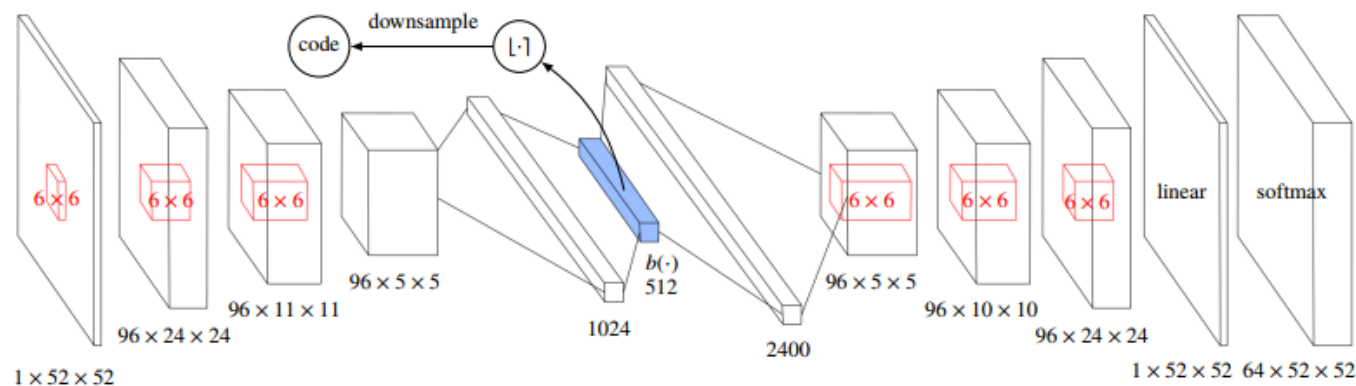
What if we still count states, but in a different space?

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

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

hang on... aren't we just checking if  $\mathbf{s} = \mathbf{s}$ ?

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

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

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!

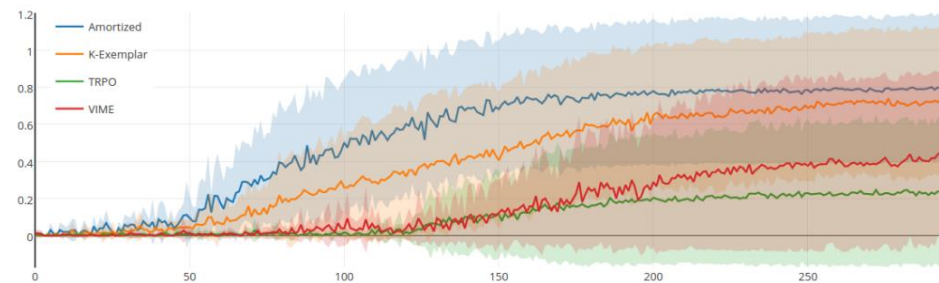
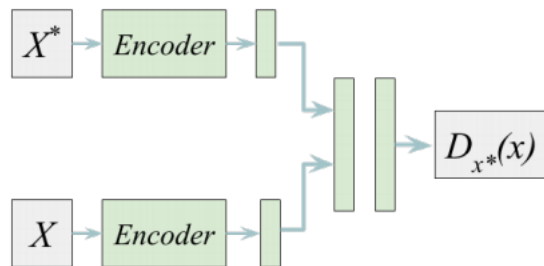


Figure 9: DoomMyWayHome+



# 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

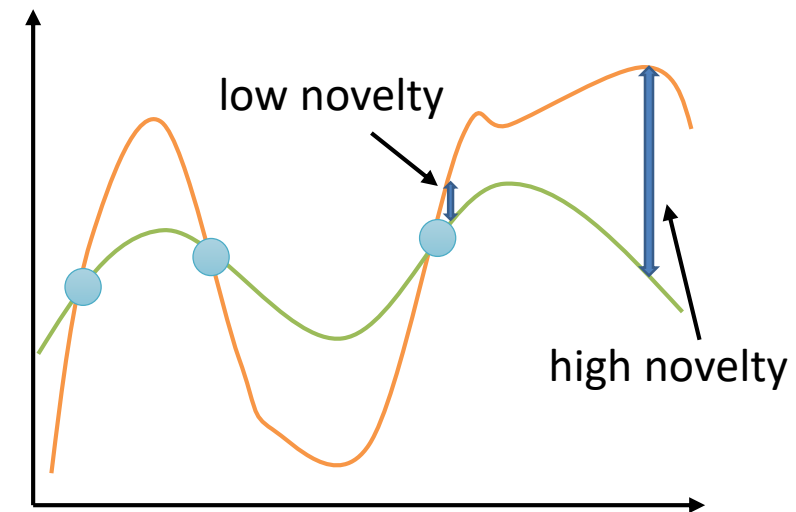
...and doesn't even need to output great densities

...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

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  $\hat{f}_\theta(\mathbf{s}, \mathbf{a})$

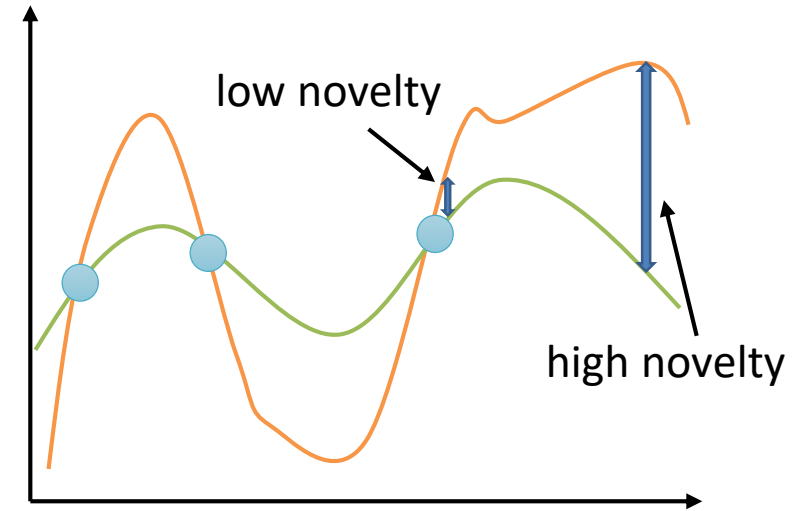
use  $\mathcal{E}(\mathbf{s}, \mathbf{a}) = \|\hat{f}_\theta(\mathbf{s}, \mathbf{a}) - f^*(\mathbf{s}, \mathbf{a})\|^2$  as bonus

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

- also related to information gain, which we'll discuss next time!

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



this will be in HW5!

# Posterior sampling in deep RL

Thompson sampling:

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

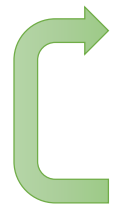
$$a = \arg \max_a E_{\theta_a}[r(a)]$$

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

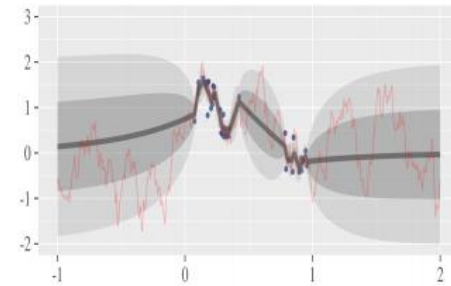
how can we represent a distribution over functions?

# Bootstrap

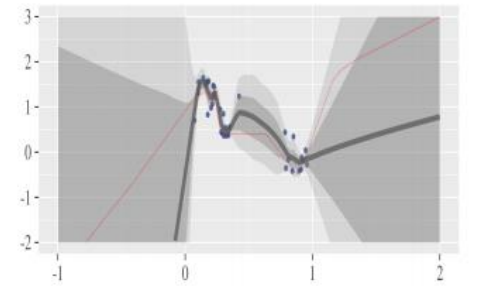
given a dataset  $\mathcal{D}$ , resample with replacement  $N$  times to get  $\mathcal{D}_1, \dots, \mathcal{D}_N$

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

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

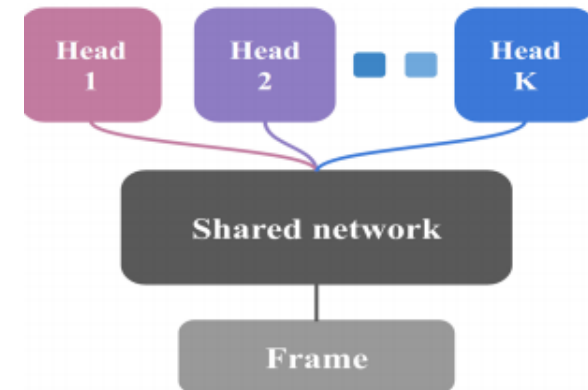


(b) Gaussian process posterior



(c) Bootstrapped neural nets

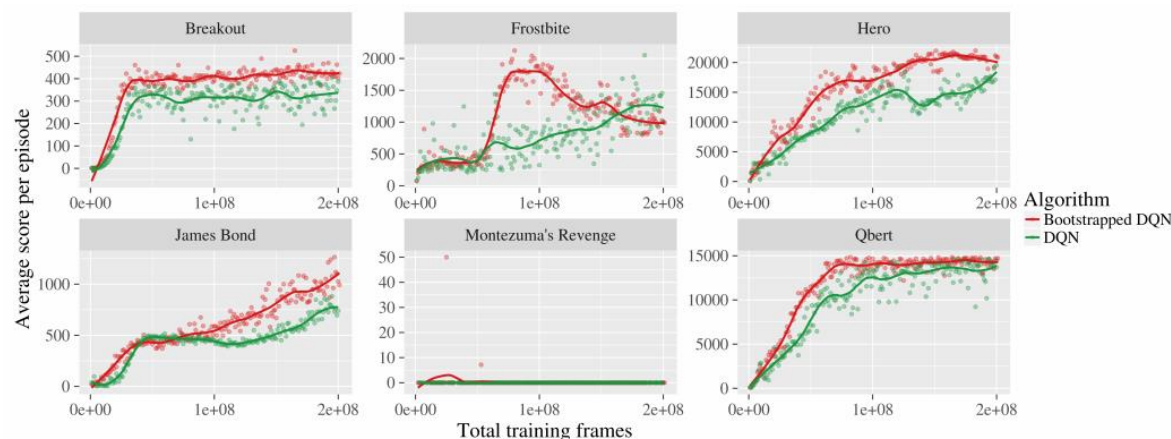
training  $N$  big neural nets is expensive, can we avoid it?



# 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