Exploration (Part 1)

CS 285

Instructor: Sergey Levine
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!

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

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

Examples from D. Silver lecture notes: http://www0.cs.ucl.ac.uk/staff/d.silver/web/Teaching_files/XX.pdf

Exploration is hard

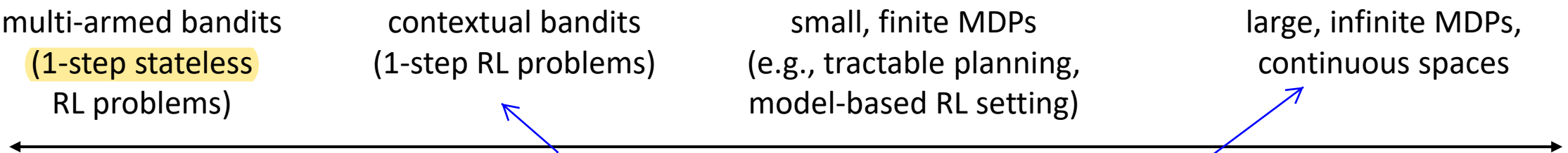
exploration is hard both practically and theoretically!

The theme in this lecture is that we show tractable and principle methods in bandits, then try to apply them to larger MDPs using some hacks.

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

what does optimal even mean?

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



theoretically tractable

theoretically tractable means that we can quantify if our exploration strategy is optimal or not

similar to multi-armed bandits expect that do have a state. But it's still only 1-step. For example, in ads, you have a feature vector about a user (state) and you have to determine which ad to show that user

we can take inspiration from the multi-armed and contextual bandits, and apply it

theoretically intractable

theoretically intractable means that we can NOT quantify if our exploration strategy is optimal or not

What makes an exploration problem tractable?

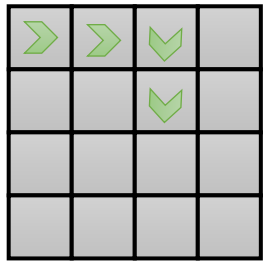
you can formalize the exploration problem as a POMDP. While multi-armed bandit is a single-step problem, we can view it as a multi-step problem, because although our actions don't affect the state, they do affect what we know. So if we explicitly demonstrate our beliefs over time, this can be represented as a temporal problem

multi-arm bandits

contextual bandits

can formalize exploration as POMDP identification

policy learning is trivial even with POMDP



small, finite MDPs

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

large or infinite MDPs

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

multi-armed bandit: now you have multiple machines, and you have decide which machine to play. Different machines will have different payouts and different reward distributions. Again, this is stochastic and we don't know the reward distribution for each arm.

one-armed bandit: you have one action and that's to pull them arm.

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

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

you don't know what the reward for pulling the arm is, but it's stochastic, so it's really a reward distribution

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

$$r(a_n) = ?$$

the reward of each arm is a probability distribution

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

unknown *per-action* reward distribution!

How can we define the bandit?

the reward for each action is distributed according to some distribution

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

we don't know theta, but we can assume we know some prior

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

so we can think of the state as our belief of each theta i

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

this, the probability distribution over our thetas, is our belief. and we update our belief every time we pull an arm. So we update the theta corresponding to that arm

how do we measure goodness of exploration algorithm?

regret: difference from optimal policy at time step T:

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

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

actual reward of action actually taken

- 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

the belief state is not the vector of thetas, but rather the probability distribution of our thetas

we can do well without solving the POMDP. doing "well" is calculated using regret

Three Classes of Exploration Methods

we will consider theoretically principled methods;
strategies that get theoretically good regret

How can we beat the bandit?

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

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

actual reward of action
actually taken

- Variety of relatively simple strategies

there are variety of relatively simple strategies that get optimal regret in the big-O sense

- 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

if you just want to exploit, you pick the action that has the largest current average reward.

many of the practical exploration strategies we use for deep RL build on this

this is a "model-free" approach in that we don't try to estimate the uncertainty. Instead we just count how many times we've selected an action. This is simple, is theoretically optimal, but is not the best in practice.

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

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

optimistic estimate: $a = \arg\max \hat{\mu}_a + C\sigma_a$

some sort of variance estimate

so we will select an action that either has a really high mean, or that has a large standard deviation, meaning that it's reward distribution is really uncertain! So we're saying "I guess any action that we haven't learned thoroughly might be good, so let's try it".

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

so this scales inversely with the number of times you select that action. so this bonus decreases with the number of times you select the action. Having T (the number of timesteps) in the numerator, ensures you explore less and less as time goes on

$\text{Reg}(T)$ 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 $\mathbf{s} = [\theta_1, \ldots, \theta_n]$

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

this is a *model* of our bandit

this is a distribution over posible bandits that we think we might have

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

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

take the optimal action

update the model

sample a vector of thetas from our belief and pretend that it's the true MDP

you'll either confirm or reject your beliefs. So your beliefs will change

- 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?

measures uncertainty

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

y could, for example, be the reward that we observed

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

information gain about z from y

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

so we want to do things that result in y's, for which the conditional entropy of z given y is as low as possible

the problem is that we don't know which y we're going to observe

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

we want this to depend on the action, so we can have information gain about z from y given our action

Information gain example

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

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"

the variable we observe is the reward for action a

the variable we want to learn is theta a, which is the parameters of the reward distribution for that action a

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

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

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

under our current belief about the MDP, what's the difference between the optimal action and the action we're currently considering. know that we don't know a* so this is just an expectation

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

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

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

General themes

UCB: optimistic

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

Thompson sampling:

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

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

you maintain a belief over theta, you sample from that belief, and then take the optimal action according to that sample

Info gain:

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

you estimate the information gain about z, based on some observation why given action 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

Exploration in Deep RL

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

"exploration bonus"

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

the intuition in RL is that we're going to select an exploration bonus for different actions AND different states. Don't worry about the square root or the 2lnT, the important thing is that it scales with the inverse of N(a)

can we use this idea with MDPs?

count the number of times we've visited a state-action tuple, or just the state

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

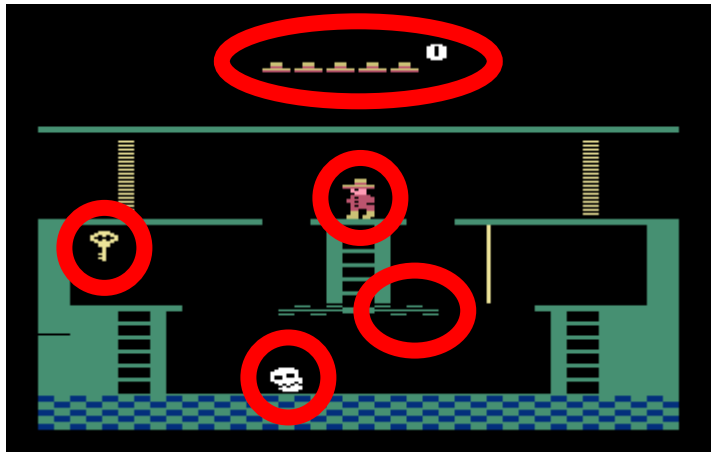
you need to figure out how to do the counting!

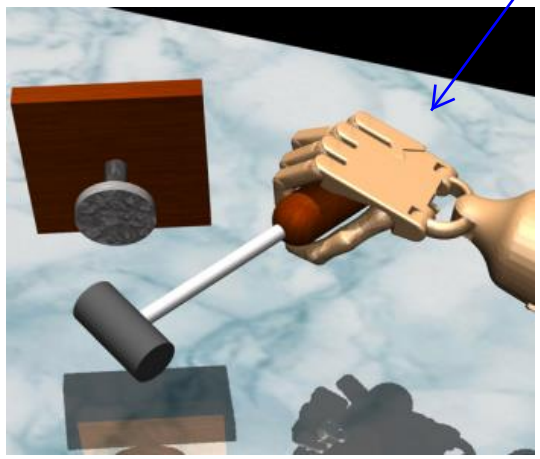
The trouble with counts

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

while it makes sense in small, discrete MDPs, it doesn't necessarily make sense in more complex MDPs

But wait… what's a count?





what about continuous spaces. So no two states will be the same!

Uh oh… we never see the same thing twice!

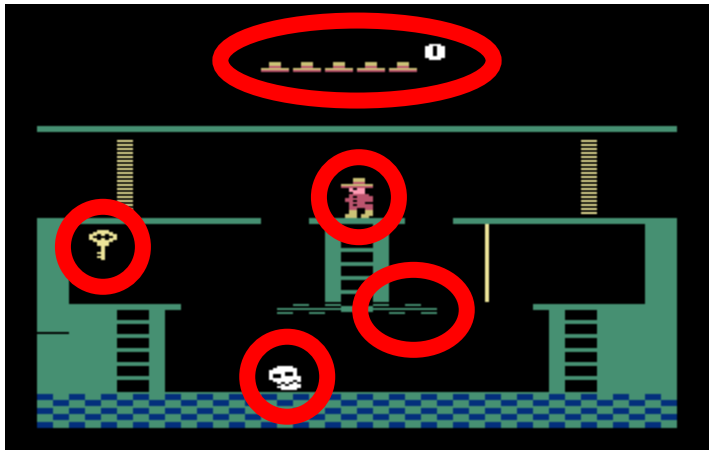
which makes counting useless

But some states are more similar than others

Fitting generative models

a density model could be something simple like a gaussian, or something more complicated like a NN

so we want to find the probability of a state or (s,a) tuple

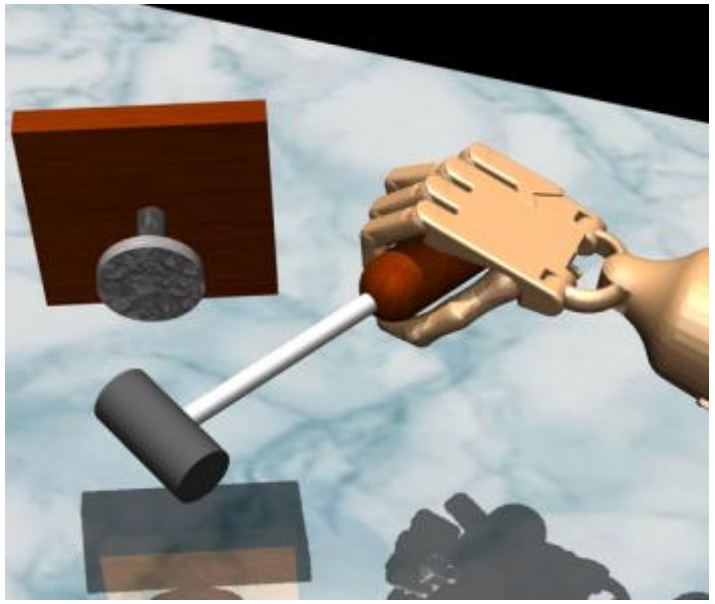


idea: fit a density model $p_\theta(\mathbf{s})$ (or $p_\theta(\mathbf{s}, \mathbf{a})$)

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

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

$N(\mathbf{s})$: count

$P(\mathbf{s})$: probability/density

n: total states visited

not total unique states, but rather just the total number of states or steps you've taken

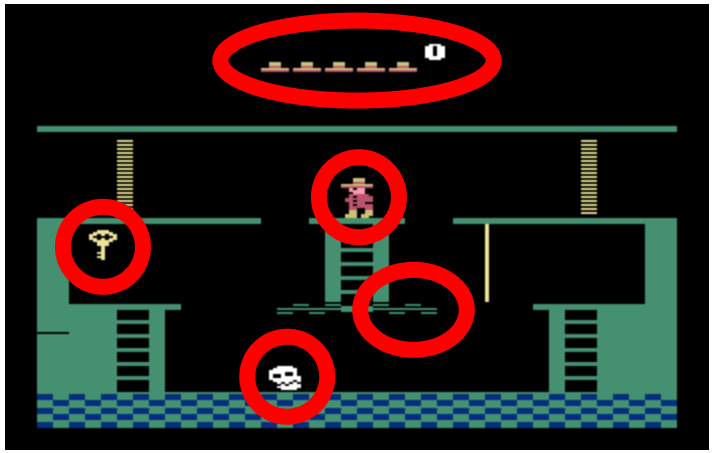
after we see $\mathbf{s}$, we have:

now we update our probabilities

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

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

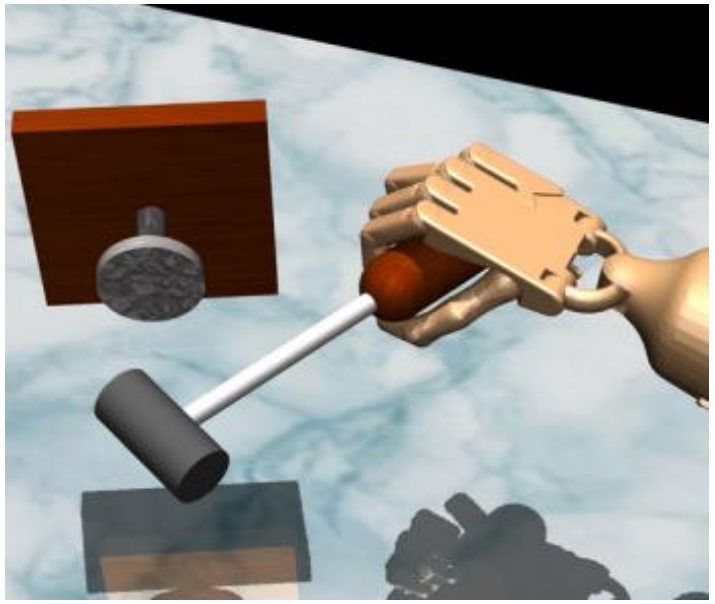
take a step i and observe $\mathbf{s}_i$

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"

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…"

What kind of bonus to use?

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

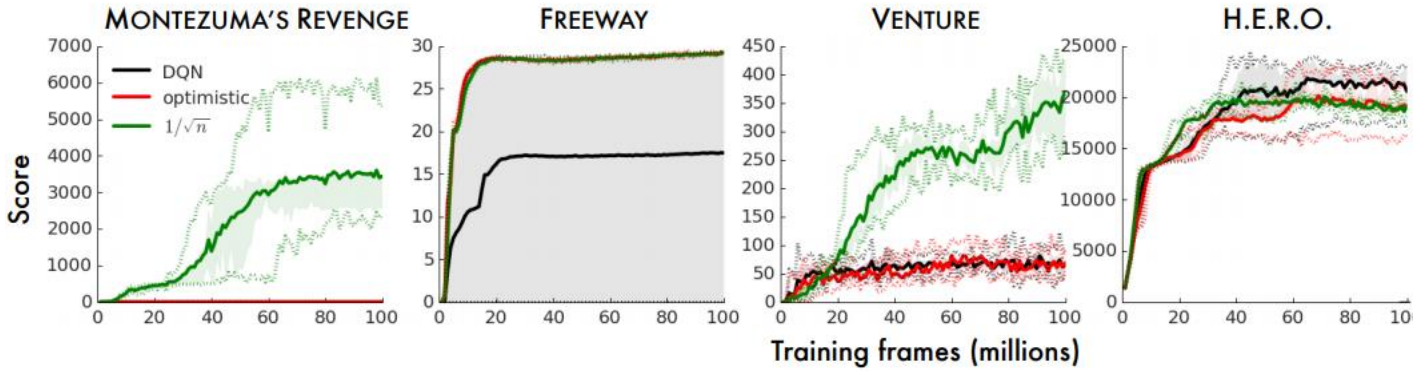
UCB: $$\mathcal{B}(N(\mathbf{s})) = \sqrt{\frac{2 \ln n}{N(\mathbf{s})}}$$

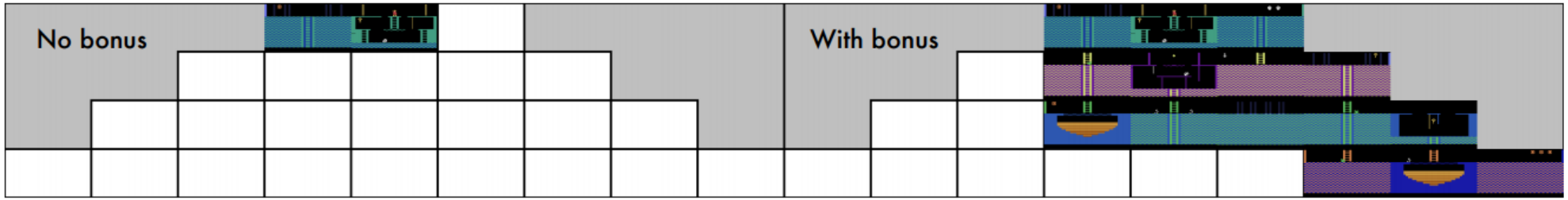
MBIE-EB (Strehl & Littman, 2008): $$\mathcal{B}(N(\mathbf{s})) = \sqrt{\frac{1}{N(\mathbf{s})}}$$

this is the one used by Bellemare et al. ‘16

BEB (Kolter & Ng, 2009): $$\mathcal{B}(N(\mathbf{s})) = \frac{1}{N(\mathbf{s})}$$

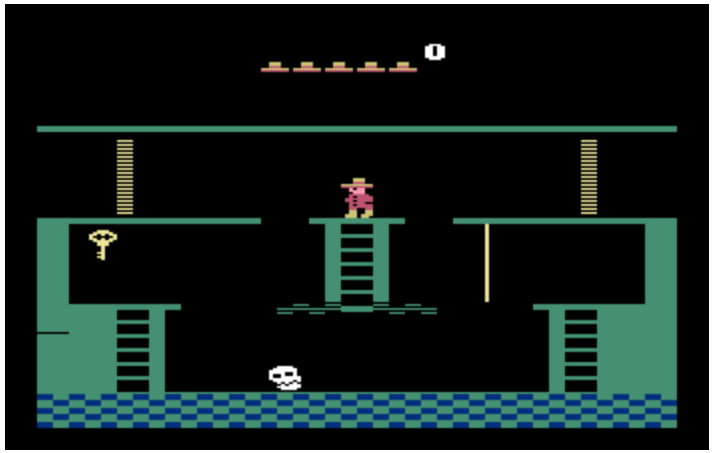
Does it work?

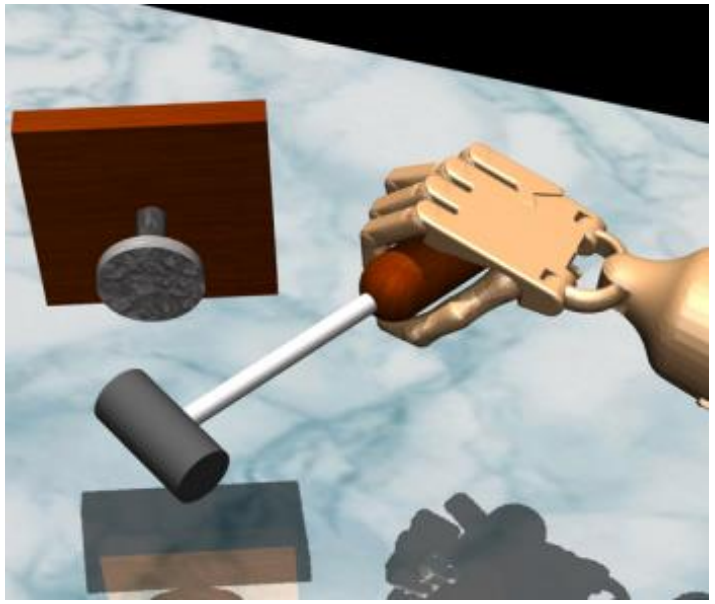




Bellemare et al. "Unifying Count-Based Exploration…"

What kind of model to use?



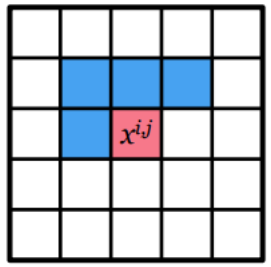


$$p_\theta(\mathbf{s})$$

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

More Novelty-Seeking Exploration

other techniques that also exploit the notion of optimism to improve exploration

Counting with hashes

count-based method with a more sophisticated density model. Count with hashes

compress s into a k-bit hashcode the encoder, phi(s). Then count the times you see the same code, instead of the same state

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

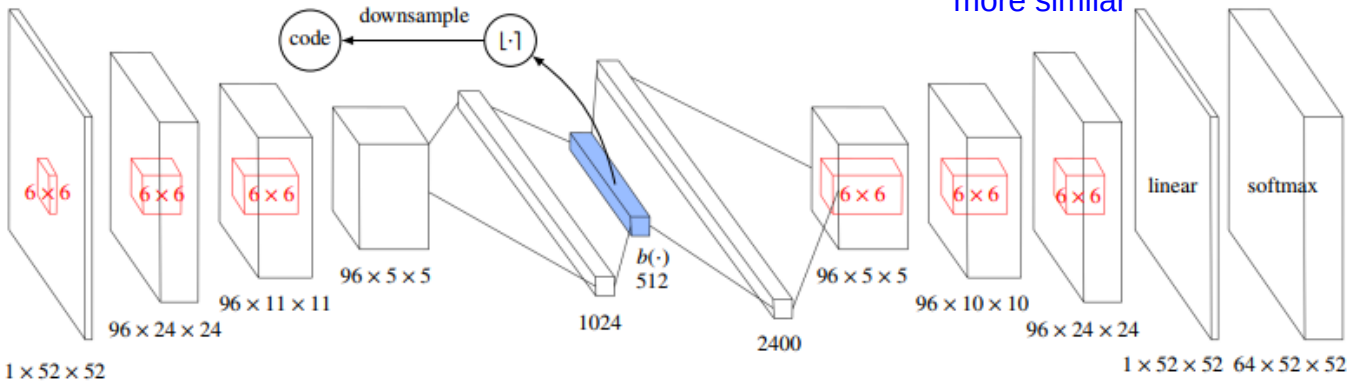
idea: compress $\mathbf{s}$ into a k-bit code via $\phi(\mathbf{s})$, then count $N(\phi(\mathbf{s}))$

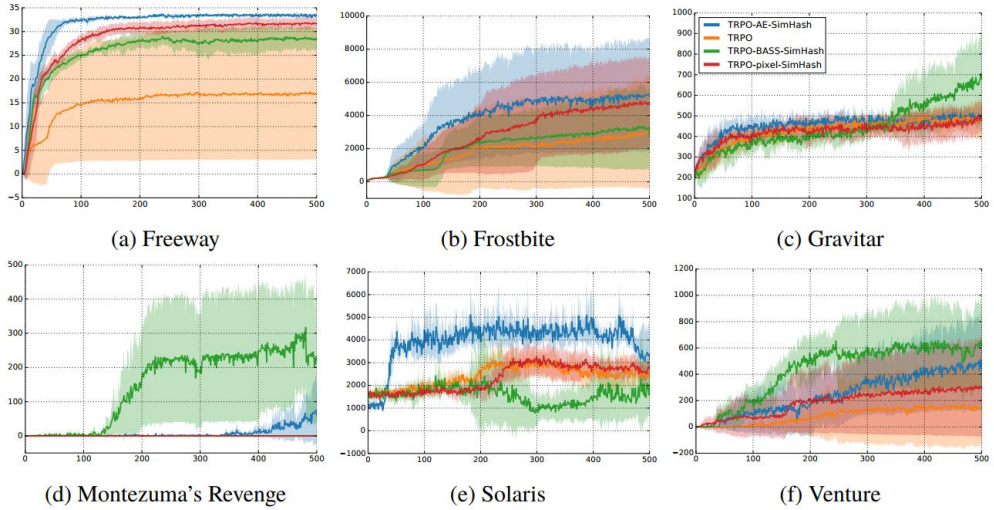
shorter codes = more hash collisions

similar states get the same hash? maybe

improve the odds by *learning* a compression:

so the way you can improve the odds of having similar states get the same hash, instead of using a hash function that seeks to minimize collisions, use an autoencoder that is trained to get the maximum reconstruction accuracy. By doing this, the collisions will be from states that are more similar





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

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

Tang et al. "#Exploration: A Study of Count-Based Exploration"

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

avoid density modeling all together by exploiting classifiers to give you density scores.

we can exploit this by devising a class of models that are particularly easy to train that can't produce samples at all, but can produce good densities

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

in the buffer

we use classifier likelihood or 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: $\mathcal{D}$

"positive" meaning that s is a new state

we compare the new state to past states. if the classifier can easily distinguish the new state from past states, then the new state is very novel and will have low density. If it can't distinguish, then we can assume the new state is very similar to old states, so it gets high density

D_s(s) is the probability that the classifier assigns to the state being a new state. The larger the count of s, the lower D_s(s) will be

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

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: $D^{\star}_{\mathbf{s}}(\mathbf{s}) = \dfrac{1}{1 + p(\mathbf{s})}$

we used something like "weight-decayed" regularizer classifier

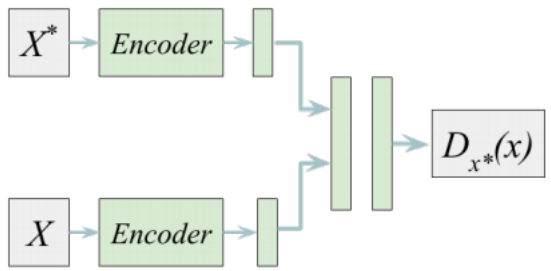
$$p_\theta(\mathbf{s}) = \frac{1 - D_{\mathbf{s}}(\mathbf{s})}{D_{\mathbf{s}}(\mathbf{s})}$$

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!

the exemplar is X*



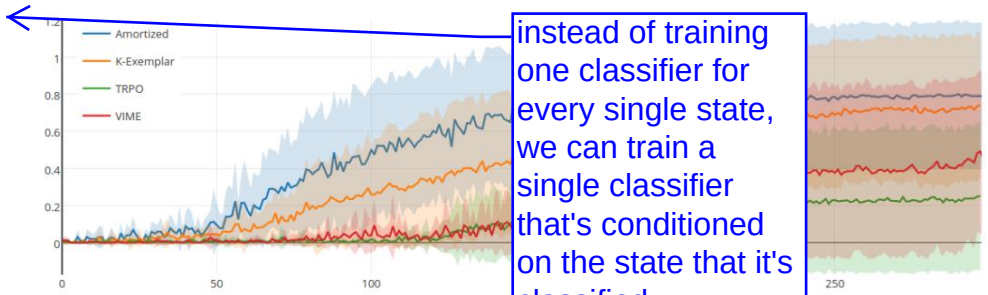


Figure 9: Doom

instead of training one classifier for every single state, we can train a single classifier that's conditioned on the state that it's classified

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

Heuristic estimation of counts via errors

there are also heuristic techniques that we can use to estimate quantities that aren't really counts, but serve as similar roles as counts and work pretty well in practice

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

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!

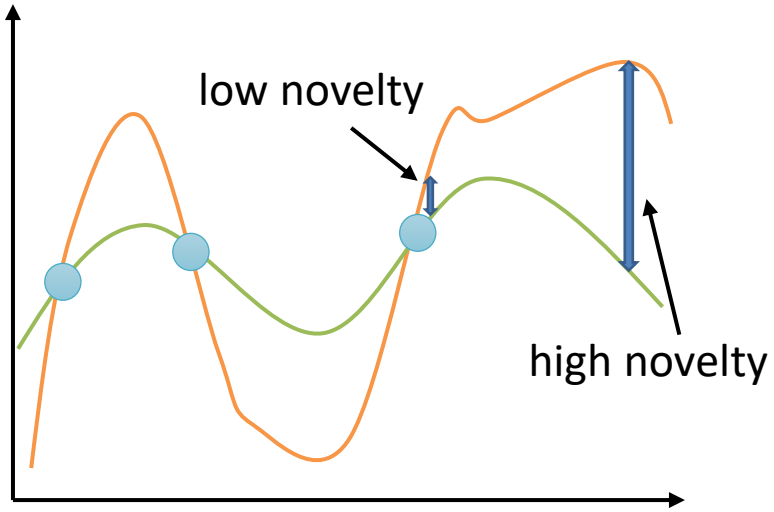
we just need some type of number or score that is larger for non-novel states and lower for novel states. The score doesn't actually have to even be the density

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

error is small for (s,a) that we've seen, and larger for (s,a) that we haven't seen!

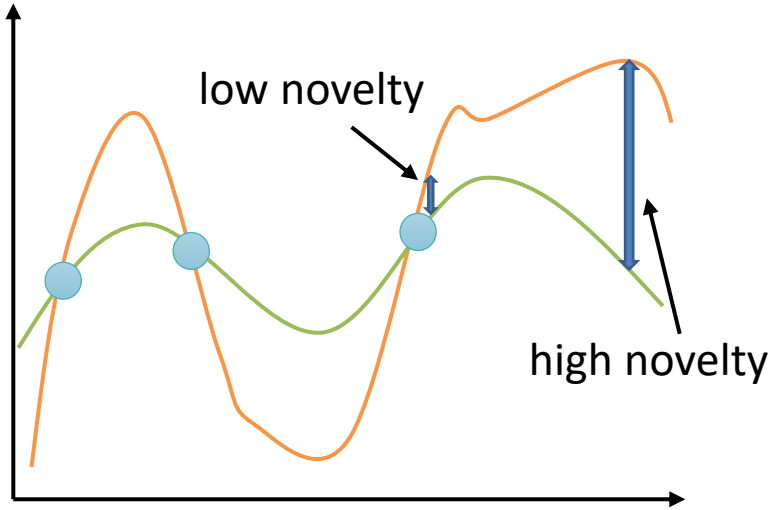


Heuristic estimation of counts via errors

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



the dynamics

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

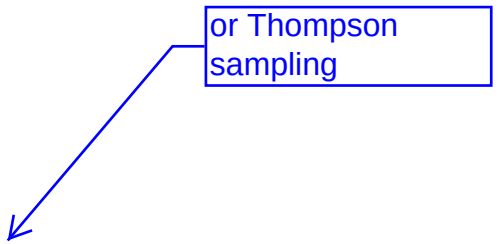
- also related to information gain, which we'll discuss next time!

this will be in HW5!

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

set f* to be a neural network with parameters phi, where phi is initialized randomly. And this network isn't actually even trained, it's an arbitrary but structured function. The point is that we don't need f* to be meaningful, we just need it to be something that can serve as a target that varies over the state and action space in ways that aren't trivial to model.

Burda et al. Exploration by random network distillation. 2018.



Posterior Sampling in Deep RL

Posterior sampling in deep RL

we estimate a model of our bandit...theta parameterizes the distribution over the bandit's rewards....so we maintain a belief over theta. At each step of exploration, we sample thetas from the beliefs, and take an action that is the argmax of the bandits described by that corresponding model

Thompson sampling:

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

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

What do we sample?

How do we represent the distribution?

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

MDP analog is the Q-function!

in a bandit setting we choose the argmax over the reward because it's a one-timestep scenario. In RL, we choose the argmax over the Q-function.

sample a q-function from a distribution of q-functions

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

note we keep the same Q for an entire episode!

how can we represent a distribution over functions?

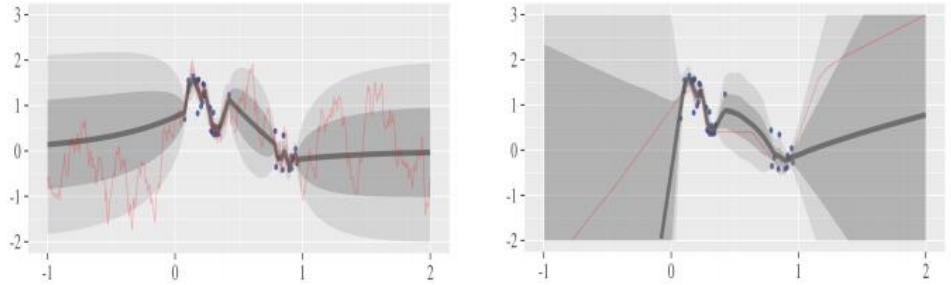
Osband et al. "Deep Exploration via Bootstrapped DQN"

Bootstrap

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

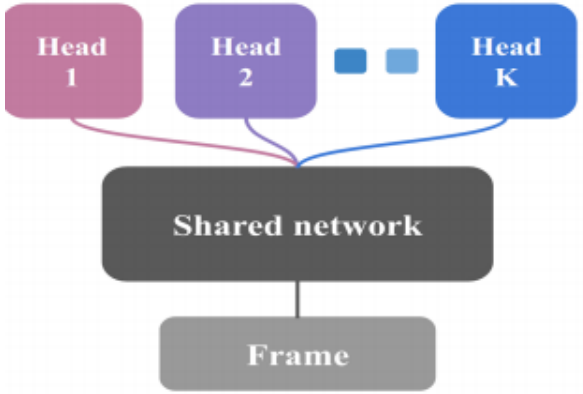
train each model f_{θ_i} on $\mathcal{D}_i$

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



(b) Gaussian process posterior (c) Bootstrapped neural nets

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



we could train one neural network with multiple heads. This means that all the layers are shared except for the last one

Osband et al. “Deep Exploration via Bootstrapped DQN”

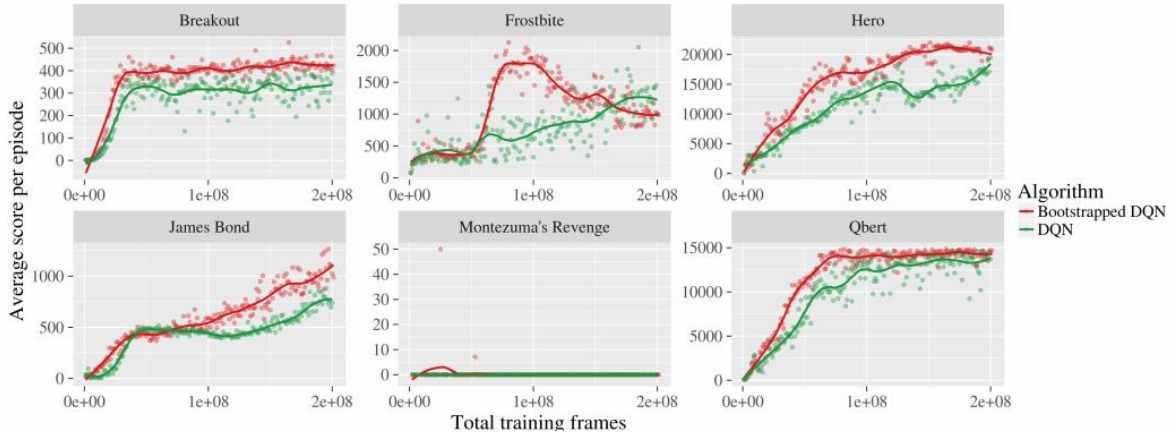
Why does this work?

in this game, you need to surface to get oxygen. So if you're exploring randomly and you're at the bottom of the ocean, it's highly unlikely you will randomly select actions that get you to the surface. This makes surfacing for air hard to discover.

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

one Q-function may decide going deeper is good, and another may decide going up is good. So if you randomly decide to select the Q-function that says going up is good, then you'll actually go up



+ no change to original reward function

- very good bonuses often do better

in practice this isn't used very often because if you have a difficult exploration problem, assigning bonuses is usually better

Osband et al. “Deep Exploration via Bootstrapped DQN”

Information Gain in Deep RL

Reasoning about information gain (approximately)

Info gain: $\mathrm{IG}(z, y|a)$

we want to select an action that, in expectation, will give us the most information about a variable of interest, z. The question to answer is: information gain about what?

information gain about *what*?

information gain about reward $r(\mathbf{s}, \mathbf{a})$?

not very useful if reward is sparse

to get IG about the state density means that you'll do things that change the state density, so you'll do novel things

can be the case in hard exploration problems where the reward is often 0

state density $p(\mathbf{s})$?

a bit strange, but somewhat makes sense!

information gain about dynamics $p(\mathbf{s}'|\mathbf{s}, \mathbf{a})$?

good proxy for *learning* the MDP, though still heuristic

if the rewards are sparse, the main thing to learn is the dynamics of the MDP

Generally intractable to use exactly, regardless of what is being estimated!

so if we want to use IG for exploration, we'll have to use some type of approximation

Reasoning about information gain (approximately)

Generally intractable to use exactly, regardless of what is being estimated

A few approximations:

this is a crude approx of IG

theta prime denotes the new parameters of a density model that we get from updating on the latest state. So if we compare the log prob of the new state before and after updating on it, that's prediction gain

prediction gain: $\log p_{\theta'}(\mathbf{s}) - \log p_{\theta}(\mathbf{s})$ (Schmidhuber '91, Bellemare '16)

intuition: if density changed a lot, the state was novel

another approx we can do is variational inference

variational inference: (Houthooft et al. "VIME")

IG can be equivalently written as $D_{\mathrm{KL}}(p(z|y)\|p(z))$

learn about *transitions* $p_\theta(s_{t+1}|s_t, a_t)$: $z = \theta$

we are going to learn about the dynamics. so z is equal to the parameters of our dynamics model

$y = (s_t, a_t, s_{t+1})$

y is a transition. So the question we are asking is, which actions should we take to get the most informative transitions about theta.

h denotes our replay buffer without our latest transition

$$D_{\mathrm{KL}}(p(\theta|h, s_t, a_t, s_{t+1})\|p(\theta|h))$$

model parameters for $p_\theta(s_{t+1}|s_t, a_t)$

history of all prior transitions

newly observed transition

intuition: a transition is more informative if it causes belief over θ to change

idea: use variational inference to estimate $q(\theta|\phi) \approx p(\theta|h)$

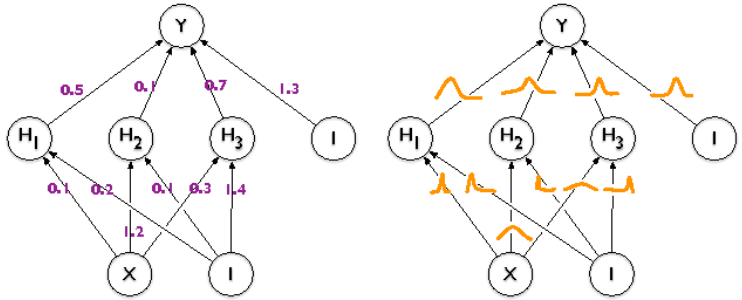
approximate posterior with variation params phi

given new transition (s, a, s'), update ϕ to get ϕ'

so we want our model parameters after observing a new transition to be very different from the model parameters from only observing the history without that transition.

Reasoning about information gain (approximately)

VIME implementation:



IG can be equivalently written as $D_{\mathrm{KL}}(p(\theta|h, s_t, a_t, s_{t+1})\|p(\theta|h))$

model parameters for $p_\theta(s_{t+1}|s_t, a_t)$

history of all prior transitions

newly observed transition

train the approximate posterior to...

$q(\theta|\phi) \approx p(\theta|h)$ specifically, optimize variational lower bound $D_{\mathrm{KL}}(q(\theta|\phi)\|p(h|\theta)p(\theta))$

represent $q(\theta|\phi)$ as product of independent Gaussian parameter distributions

with mean ϕ (see Blundell et al. "Weight uncertainty in neural networks")

given new transition (s, a, s'), update ϕ to get ϕ'

i.e., update the network weight means and variances

use $D_{\mathrm{KL}}(q(\theta|\phi')\|q(\theta|\phi))$ as approximate bonus

$$p(\theta|\mathcal{D}) = \prod_i p(\theta_i|\mathcal{D})$$

$$p(\theta_i|\mathcal{D}) = \mathcal{N}(\mu_i, \sigma_i)$$

ϕ

pretty much this is saying that we represent each parameter phi_i as a gaussian

the higher the change in parameters, the larger the bonus

Houthooft et al. "VIME"

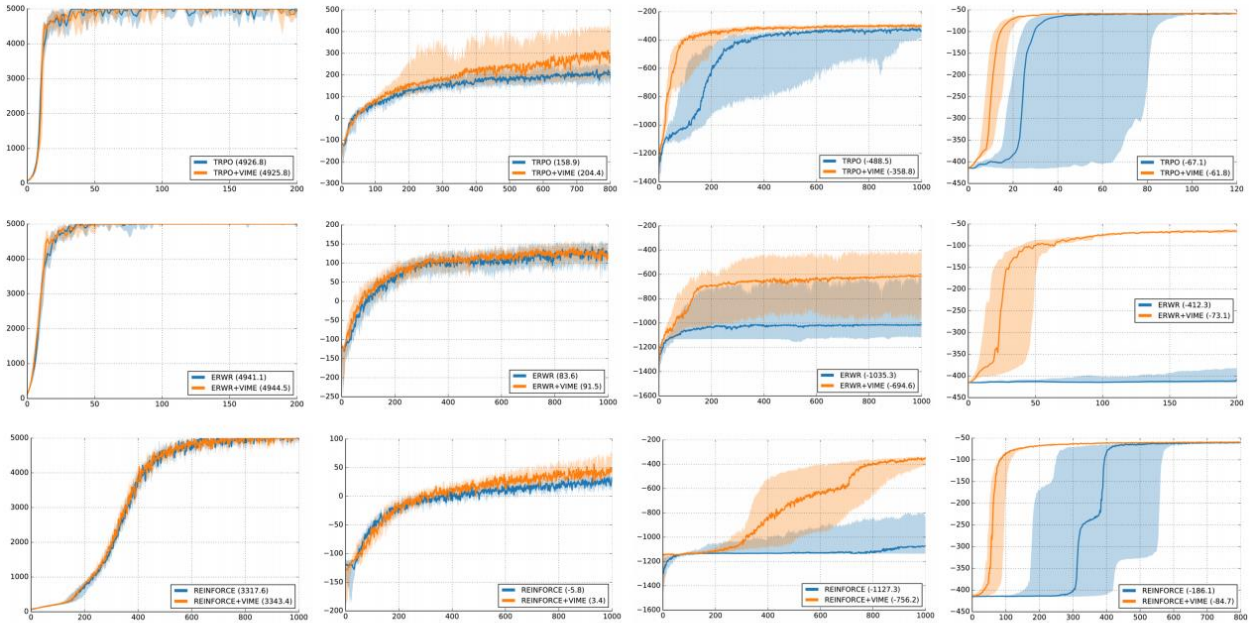
Reasoning about information gain (approximately)

VIME implementation:

IG can be equivalently written as $D_{\mathrm{KL}}(p(\theta|h, s_t, a_t, s_{t+1})\|p(\theta|h))$

$q(\theta|\phi) \approx p(\theta|h)$ specifically, optimize variational lower bound $D_{\mathrm{KL}}(q(\theta|\phi)\|p(h|\theta)p(\theta))$

use $D_{\mathrm{KL}}(q(\theta|\phi')\|q(\theta|\phi))$ as approximate bonus



(a) CartPole (b) CartPoleSwingup (c) DoublePendulum (d) MountainCar

Approximate IG:

+ appealing mathematical formalism

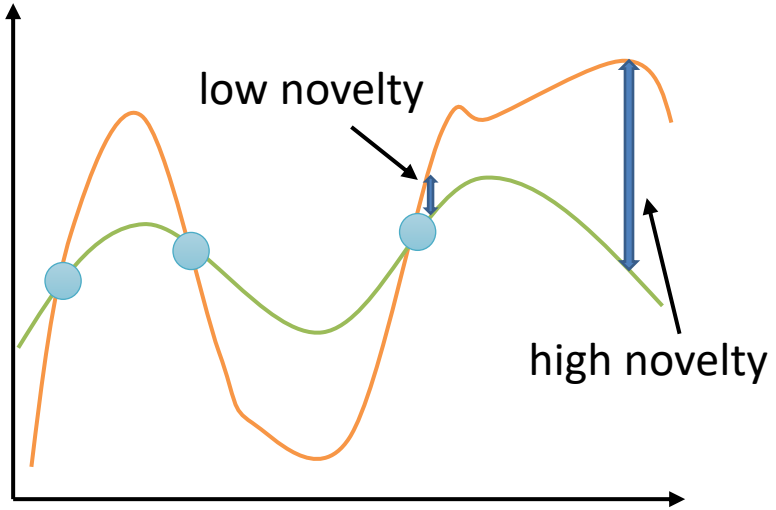
- models are more complex, generally harder to use effectively

Houthooft et al. “VIME”

Exploration with model errors

$D_{\mathrm{KL}}(q(\theta|\phi')\|q(\theta|\phi))$ can be seen as change in network (mean) parameters ϕ

if we forget about IG, there are many other ways to measure this



Stadie et al. 2015:

- encode image observations using auto-encoder
- build predictive model on auto-encoder latent states
- use model error as exploration bonus

Schmidhuber et al. (see, e.g. “Formal Theory of Creativity, Fun, and Intrinsic Motivation):

- exploration bonus for model error
- exploration bonus for model gradient
- many other variations

Many others!

Recap: classes of exploration methods in deep RL

- Optimistic exploration:
  - Exploration with counts and pseudo-counts
  - Different models for estimating densities
- Thompson sampling style algorithms:
  - Maintain a distribution over models via bootstrapping
  - Distribution over Q-functions
- Information gain style algorithms
  - Generally intractable
  - Can use variational approximation to information gain

Suggested readings

Schmidhuber. (1992). **A Possibility for Implementing Curiosity and Boredom in Model-Building Neural Controllers.**

Stadie, Levine, Abbeel (2015). **Incentivizing Exploration in Reinforcement Learning with Deep Predictive Models.**

Osband, Blundell, Pritzel, Van Roy. (2016). **Deep Exploration via Bootstrapped DQN.**

Houthooft, Chen, Duan, Schulman, De Turck, Abbeel. (2016). **VIME: Variational Information Maximizing Exploration.**

Bellemare, Srinivasan, Ostroviski, Schaul, Saxton, Munos. (2016). **Unifying Count-Based Exploration and Intrinsic Motivation.**

Tang, Houthooft, Foote, Stooke, Chen, Duan, Schulman, De Turck, Abbeel. (2016). **#Exploration: A Study of Count-Based Exploration for Deep Reinforcement Learning.**

Fu, Co-Reyes, Levine. (2017). **EX2: Exploration with Exemplar Models for Deep Reinforcement Learning.**