Exploration (Part 1)

CS 294-112: Deep Reinforcement Learning
Sergey Levine
Class Notes

1. Homework 4 due next Wednesday!
2. Project milestone report due in two weeks!
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

• “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

Mao

• 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
  • \textbf{Exploitation}: go to your favorite restaurant
  • \textbf{Exploration}: try a new restaurant

• Online ad placement
  • \textbf{Exploitation}: show the most successful advertisement
  • \textbf{Exploration}: show a different random advertisement

• Oil drilling
  • \textbf{Exploitation}: drill at the best known location
  • \textbf{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

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)
- Theoretically tractable

**Contextual bandits**
- (1-step RL problems)
- Theoretically intractable

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

**Large, infinite MDPs**
- Continuous spaces
- 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

\[ A = \{\text{pull arm}\} \]
\[ r(\text{pull arm}) =? \]

\[ A = \{\text{pull}_1, \text{pull}_2, \ldots, \text{pull}_n\} \]
\[ r(\text{a}_n) =? \]
assume \[ r(\text{a}_n) \sim p(\text{r}|\text{a}_n) \]
unknown per-action reward distribution!
Let’s play!

- 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?

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 \( s = [\theta_1, \ldots, \theta_n] \)

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

- 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?

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

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

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):

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

Reg($T$) is $O(\log T)$, provably as good as any algorithm
Probability matching/posterior sampling

\[ r(a_i) \sim p_{\theta_i}(r_i) \]

this defines a POMDP with \( s = [\theta_1, \ldots, \theta_n] \)

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

\( \text{this is a model of our bandit} \)

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

- This is called posterior sampling or Thompson sampling
- Harder to analyze theoretically
- Can work very well empirically

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

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

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”

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

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

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

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:**

\[ \theta_1, \ldots, \theta_n \sim \hat{\theta}(\theta_1, \ldots, \theta_n) \]

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

**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: \[ a = \operatorname{arg\,max} \mu_a + \sqrt{\frac{2 \ln T}{N(a)}} \]

“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 \( N(s, a) \) or \( N(s) \) to add exploration bonus

use \( r^+(s, a) = r(s, a) + B(N(s)) \)

Bonus that decreases with \( N(s) \)

Use \( r^+(s, a) \) instead of \( r(s, a) \) with any model-free algorithm

+ simple addition to any RL algorithm

- need to tune bonus weight
The trouble with counts

use $r^+(s, a) = r(s, a) + B(N(s))$

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 $p_\theta(s)$ (or $p_\theta(s, a)$)

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

if $s$ is similar to previously seen states

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

if we have small MDPs

the true probability is:

\[
P(s) = \frac{N(s)}{n}
\]

probability/density

total states visited

after we see $s$, we have:

\[
P'(s) = \frac{N(s) + 1}{n + 1}
\]

count

total states visited

can we get $p_\theta(s)$ and $p_\theta'(s)$ to obey these equations?
Exploring with pseudo-counts

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

take a step $i$ and observe $s_i$

fit new model $p_{\theta'}(s)$ to $\mathcal{D} \cup s_i$

use $p_\theta(s_i)$ and $p_{\theta'}(s_i)$ to estimate $\hat{N}(s)$

set $r_i^+ = r_i + B(\hat{N}(s))$ “pseudo-count”

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

$$p_\theta(s_i) = \frac{\hat{N}(s_i)}{\hat{n}}$$

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

two equations and two unknowns!

$$\hat{N}(s_i) = \hat{n} p_\theta(s_i)$$

$$\hat{n} = \frac{1 - p_{\theta'}(s_i)}{p_{\theta'}(s_i) - p_\theta(s_i) p_\theta(s_i)}$$

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:**

\[ B(N(s)) = \sqrt{\frac{2 \ln n}{N(s)}} \]

**MBIE-EB (Strehl & Littman, 2008):**

\[ B(N(s)) = \sqrt{\frac{1}{N(s)}} \]

**BEB (Kolter & Ng, 2009):**

\[ B(N(s)) = \frac{1}{N(s)} \]

this is the one used by Bellemare et al. ‘16
Does it work?

Bellemare et al. “Unifying Count-Based Exploration...”
What kind of model to use?

\[ p_\theta(s) \]

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 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

\[ p_\theta(s) \] 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 \( s \), fit a classifier to classify that state against all past states \( D \), use classifier error to obtain density

\[
p_\theta(s) = \frac{1 - D_s(s)}{D_s(s)} \quad \text{probability that classifier assigns that } s \text{ is “positive”}
\]

positives: \{s\}
negatives: \( D \)

Fu et al. “EX2: Exploration with Exemplar Models...”
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: \[ D_s^*(s) = \frac{1}{1 + p(s)} \]

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!

Fu et al. “EX2: Exploration with Exemplar Models...”
Posterior sampling in deep RL

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?

since \( Q \)-learning is off-policy, we don’t care which \( Q \)-function was used to collect data

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

MDP analog is the \( Q \)-function!

1. sample \( Q \)-function \( \hat{Q} \) from \( p(Q) \)
2. act according to \( \hat{Q} \) for one episode
3. update \( p(Q) \)

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 $f_{\theta_i}$ on $\mathcal{D}_i$

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

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

Osband et al. “Deep Exploration via Bootstrapped DQN”
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”