Extended Abstract

Intrinsic motivation in reinforcement learning has seen many successes in the past, from achieving higher performance on atari games to allowing for robots to learn diverse sets of skills. However, augmenting the reward of a MDP can lead to another source of instability for the algorithm, causing it to chase a moving reward. By instead conditioning the reward by the goal and amoritizing the task of exploration onto choosing appropriate goals, we avoid the instabilities of count-based methods

In this paper, we consider how to efficiently explore an environment through a goal-conditioned MDP. We that by decoupling deep RL policies trained with intrinsic motivation vs. extrinsic reward, we would be able to achieve better, more directed exploration than exploration methods that combine the two. To provide more information about our method, we define a Markov Decision Process (S, A, T, r, γ) to solve. In order to use goal-directed exploration, we will then use a new goal-conditioned MDP (S, A, T, r, γ, G) featuring equivalent dynamics as the original MDP as well as a goal space G containing elements of S. To explore original environment, we will sample a goal $g \in G$ according to some strategy, then traverse the MDP with a policy $\pi(a|s, g)$. By choosing promising goals in the replay buffer to return to and explore from, we are able to directly explore desired parts of the state space.

In order to create a goal-sampling strategy which performs well, we utilize two assumptions about the environment. The first is that some goals are more promising than others for future exploration (i.e. the goal is in some critical region of the environment where other interesting behaviors can be learned). The second is that the dynamics of certain regions of the state space are similar to others, which will allow us to have generalized behaviors from goal conditioned policies. Basically, a skill learned from conditioning on one goal can translate to another, because the world's dynamics does not change drastically between nearby states.

We show that empowerment serves as a powerful heurstic to select goals by. Empowerment measures the controllability of a current state in an MDP. We estimate empowerment by estimating the number of reachable states originating from a certain state. By choosing goals with empowerment, we were able to get results that perform similar SkewFit. However, we acknowledge that as horizons become larger and environments become more complicated, using goal-conditioned reinforcement learning to explore may not perform well do to the inability of these methods to learn to reach far-away goals well.

GCExplore: Reaching High Empowerment States via Goal-Conditioned Exploration

Jonathan Yang, Justin Yu
Department of Computer Science
University of California, Berkeley
Berkeley, CA
jy2370@berkeley.edu, justinvyu@berkeley.edu

1 Introduction

Reinforcement learning (RL) provides a framework to model an agent's interaction with their environment in the form of maximizing expected returns corresponding to a task. A major problem in RL is exploration, which requires an agent to take diverse actions at diverse states in order to find which parts of the environment have the highest reward signal.

In the absence of shaped rewards from the environment to provide supervision, intrinsic motivation helps encourage behaviors that eventually lead to success. Forms of intrinsic motivation include count-based exploration bonuses, dynamics or value prediction error, and maximizing empowerment, among many others. Oftentimes, intrinsic motivation is added to the reward signal as a bonus, which incentivizes an agent to visit new states when it exploits its current policy. However, this leads to another source of instability for reinforcement learning algorithms, causing the policy to chase a moving reward.

In this paper, we use goal-conditioned reinforcement learning in order to direct an agent to explore new states without augmenting the reward function. By choosing promising goals in the replay buffer to return to and explore from, we are able to directly explore desired parts of the state space. In addition, we propose that empowerment is a viable goal-selection strategy. By choosing goals which have the most controllability in the future, we can return to areas in the environment that enable the most future exploration.

2 Background

2.1 Reinforcement Learning and Markov Decision Processes

Reinforcement learning aims to learn a policy π that solves a Markov Decision Process defined by (S, A, R, γ). S is the state space, A is the action space, $R : S \times A \to \mathbb{R}$ is a function that maps a state and action to a reward, and γ is the discount factor. The reinforcement learning objective is given by

$$\max \sum_{t=0}^{T} E_{(s_t,a_t)\sim p_\pi}[r(s_t, a_t)]$$

To provide more stability, we use the soft-actor critic algorithm (SAC) ([5]). This algorithm augments the reinforcement learning objective to also maximize the entropy as follows:

$$\max \sum_{t=0}^{T} E_{(s_t,a_t)\sim p_\pi}[r(s_t, a_t) + \alpha\mathcal{H}(\pi(\cdot|s_t)]$$

Preprint. Under review.

2.2 Goal Conditioned Reinforcement Learning

In goal conditioned RL, we augment a typical MDP with a goal space G and condition the policy on a representation of the goal we want it to achieve. This inherently places the agent in a multi-task setting, where conditioning on different goals leads to different behaviors. While our experiments focus on the task of "goal reaching", where the specified goal is some notion of a location that the agent should reach, the tasks corresponding to different goals can vary arbitrarily. The policy $\pi_\theta(a|s, g)$ is conditioned on the state as well as a goal representation g. To make goal-conditioned reinforcement learning more sample-efficient, we relabel goals with Hindsight Experience Replay [1].

2.3 Empowerment

Empowerment is an information theoretic measure of the control an agent has over its environment. It views the environment as a communication channel, which the agent will try transmitting "information" in the form of actions to reach a desired state in the state space. We can define state-conditional empowerment, which intuitively quantifies the amount of influence an agent has on their future, given that they start at a state s.

Formally, N-step empowerment is defined as the channel capacity between an N-step action sequence $a_t^{t+N-1} = (a_t, a_{t+1}, \ldots, a_{t+N-1})$ and the final state s_{t+N}, all conditioned on starting at a state s_t. In the following equation, capital letters correspond to random variables, and lowercase letters correspond to a value the random variable takes on.

$$\begin{aligned}\mathcal{E}(s_t) = \text{Empowerment}(s_t) &= \mathcal{C}(A_t^{t+N-1}; S_{t+N}|S_t = s_t) \\ &= \max_{\omega(A_t^{t+N-1}|S_t=s_t)} \mathcal{I}(A_t^{t+N-1}; S_{t+N}|S_t = s_t)\end{aligned}$$

In tabular settings, the N-step empowerment at a certain state corresponds to the $\log$ of the number of states reachable in N actions (cite Salge).

3 Related Work

3.1 Novelty Bonuses

Typically, in reinforcement learning, intrinsic motivation is provided by a novelty bonus to the reward function. These novelty bonuses incentivize exploration to new parts of the state space by increasing the reward for state, action pairs that have been explore less. The most common form of these bonuses comes from finding the density of the visited state action pairs, then rewarding areas with lower density. For example, Bellmare et. al. fits a density model finds computes pseudocounts for each state [2]. Tang et. al. proposes a different approach by converting high-dimensional states into hash codes and augments the reward with the inverse square root of the empirical count of a state's hash code [8]. Other novelty bonuses exist that give a bonus based on how well a model has learned about an environment at a certain state and action. For example, curiosity-based methods find the an error of a forward predictive model [3],[7]. Random network distillation proposes that a good exploration bonus can be found by computing the error of a model being fit to a random neural network.

3.2 Maximizing Empowerment

Another form of intrinsic motivation is empowerment. The largest hurdle to using empowerment is that it is very difficult to compute, due to requiring a mutual information computation, which is generally intractable. Previous work has gotten around this computational intractability by optimizing a variational lower bound objective [6]. We borrow the empowerment approximation approach from [9], which fits a Gaussian channel model of the form of an affine transformation to the N-step action sequence: $G(s_t)\vec{a}_t^{t+N} + \vec{b}(s_t)$ and uses the "Water-Filling" algorithm to solve for the maximum channel capacity.

Concretely, we fit a neural network to minimize $||G(s_t)\vec{a}_t^{t+N} + \vec{b}(s_t) - s_{t+N}||_2^2$ using N-step trajectories in the replay buffer. Then, we compute the maximum channel capacity (empowerment)

given s_t by computing the SVD of $G(s_t)$ and run a convex optimization procedure to maximize the following constrained problem:

$$\max_{p_i} \frac{1}{2} \sum_{i=1}^{k} \log(1 + \sigma_i(s_t) p_i) \text{ s.t. } \sum_{i=1}^{k} p_i = 1$$

We refer the reader to [9] for a more detailed explanation of the setup.

3.3 Other Exploration Schemes

There has been past work on exploring by directly visiting promising states. The idea that we drew parts of our method from is Go-Explore, which hypothesizes that a shortcoming of many count-based exploration methods is detachment, where an algorithm forgets a promising area it visited [4]. It proposes a new algorithm which keeps a tabulation of which trajectories have the best scores, teleports to a cell in the archive, then explores from this cell. Go-Explore then provides a robustification step, where a policy is trained with imitation learning to learn the best trajectory. The major downfall in Go-Explore is its exploitation of a simulator, which allows the algorithm to reset itself to any state.

4 Methodology

We sketch the algorithm below, highlighting the main differences from standard RL algorithms. The algorithm has three main phases: (1) goal reaching, (2) exploration, and (3) off-policy training. **First,** we run a goal-conditioned policy to reach goals set by a (possibly) intrinsically motivated goal generator $G(R)$, drawing from the replay buffer R. **Second,** we sample an environment-defined goal, and with probability ϵ, we run a random exploration policy π_{random}, and otherwise we run the same policy as before, conditioned on the new extrinsic goal. **Third**, we sample batches from our replay

buffer, consisting of trajectories from both of the first two phases, and train our policy with off-policy RL.

Algorithm 1: GCExplore

Given:
A reward function r (e.g. sparse goal-reaching reward $r(s, a, g) = -(||s - g||_2 > \epsilon)$);
A set of environment goals G_f (e.g. $G_f = \{s_{goal}\}$);

Initialize a hindsight relabeling replay buffer $\mathcal{R}$;
Initialize an off-policy or offline RL algorithm (e.g. SAC, SAC + CQL) with access to goal-conditioned policy $\pi_\theta(a|s, g)$ and Q-function $Q_\phi(a|s, g)$;
Initialize a goal generator $G(R)$ (from choices RandomGoalGenerator, SkewFitGoalGenerator, EmpowermentGoalGenerator);
if *G(R) is EmpowermentGoalGenerator* **then**
  Initialize an N-step Gaussian channel model $\mathcal{G}(s_t, a_t^{t+N}, s_{t+N})$ to estimate empowerment;
end
Initialize a random exploration schedule $\epsilon(n)$;
Initialize a random exploration policy π_{random};
for *episode = 1* $\rightarrow M$ **do**
  Sample a goal from the goal generator $g \sim G(R)$ (e.g. sampling goals based on estimated N-step empowerment values $\mathcal{E}(g)$);
  Sample an initial state $s_0 \sim p(s_0)$;
  Starting from s_0, collect T time steps of experience into $\mathcal{R}$, sampling from π_θ;
  Sample a goal from the environment $g \sim G_f$;
  Sample a random number $\alpha \sim$ Uniform$[0, 1)$;
  if $\alpha < \epsilon(episode)$ **then**
    $\pi_{explore} = \pi_{random}$;
  else
    $\pi_{explore} = \pi_\theta$;
  end
  Starting from s_t, collect T time steps of experience into $\mathcal{R}$, sampling from $\pi_{explore}$;
  Perform hindsight relabeling by sampling reached goals and recomputing reward $r' = r(s_t, a_t, g')$;
  for t_{train} *= 1* $\rightarrow N$ **do**
    Sample a minibatch $\mathcal{B}$ from the $\mathcal{R}$;
    Update networks on the $\mathcal{B}$ using the off-policy algorithm trainer;
  end
  for $t_{train,G}$ *= 1* $\rightarrow N_G$ **do**
    Sample a minibatch $\mathcal{B}$ from $\mathcal{R}$;
    Update goal generator G on $\mathcal{B}$ (e.g. optimize the Gaussian channel model);
  end
end

4.1 Goal Generation and Reaching Phase

RandomGoalGenerator This is the simplest goal generation scheme, which samples a goal at random from the set of visited states in the replay buffer.

SkewFitGoalGenerator We use an oracle version of SkewFit as a baseline, where we sample goals based on a discretized estimate of the true state density $\rho(s)$. In particular, we use $\alpha = -1$ from the paper, which results in a uniform sampling of the set of visited states by down-weighting the chance of drawing commonly seen states by the inverse density. We tried other experiments using $\alpha = -1.2, -0.9$, where $\alpha = -1.2$ represents a more adventurous goal generator, and $\alpha = -0.9$ represents more conservative goal choosing. However, this choice of α did not yield any significant performance boosts.

EmpowermentGoalGenerator We choose goals based on empowerment in a three step procedure:

1. Sample a batch $B_{candidate}$ of size $N_{candidate}$ from the replay buffer according to the same procedure as above with a chosen α, assigning lower probability to frequently seen states.
2. From this sampled batch of candidate goals, compute the empowerment using the water-filling convex optimization procedure on the trained Gaussian channel model $\mathcal{G}$ described in Section 3.2. Normalize the empowerment values by subtracting the mean and dividing by the standard deviation to retrieve $\tilde{\mathcal{E}}(s_i)$.
3. Now, choose the goal by weighting candidates by a softmax (with an added temperature parameter T) probability distribution:

$$p(s_i) = \frac{\exp\left(\tilde{\mathcal{E}}(s_i)/T\right)}{\sum_{j=1}^{N_{candidate}} \exp\left(\tilde{\mathcal{E}}(s_j)/T\right)}$$

We now describe some of the design decisions behind this empowerment-based goal-choosing scheme. We experimented with using a simple normalized empowerment probability distribution, where $p(s_i) = \frac{\mathcal{E}(s_i)}{\sum_{j=1}^{N_{candidate}} \mathcal{E}(s_j)}$. This is possible due to channel capacity being a non-negative quantity; however, this sample distribution gave too much weight to low-empowerment points, so we looked to choosing based on a softmax as a natural alternative.

We found that the choice of the α and T hyperparameters impacted performance greatly. Tuning α determines how "in-distribution" we want to our goals to be relative to the replay buffer. Tuning T changes how "hard" the softmax distribution is, so decreasing $T > 0$ moves the sampling closer and closer to an argmax over empowerment values and how much we care about sticking to the highest empowerment states. A hyperparameter search resulted over different environments gave $\alpha = -1, T = 0.5$ as a suitable choice.

The reason why we do sample a candidate batch using SkewFit in the first place is due to the alternative of computing the empowerment of every point in the replay buffer and sampling according to this softmax distribution was computationally expensive, and it also biased parts of the state space that have been seen more.

4.2 Exploration Phase

Similar to GoExplore, our exploration phase consists of rolling out a random policy after "resetting" to a high potential state. The difference is that instead of relying on simulation resets to return to these high potential states, we use an online goal-conditioned policy, which acts as a memory to return to these states. Another difference that GCExplore incorporates is an ϵ-greedy strategy of continuing to roll out the goal-reaching policy π_{theta} with probability $1 - \epsilon$ or switching to the random policy. We found that setting this parameter ϵ on a decreasing schedule was important to solve some of the tasks, since keeping the random policy forever presented learning challenges with lots of noisy relabeled data. A constant schedule of $\epsilon(n) = 0.5$ and a linear schedule of $\epsilon(n) = 1 \rightarrow 0.1$ over 50,000 steps worked well.

4.3 Training with Off-policy RL

Lastly, we train our goal-conditioned policy with an off-policy RL algorithm such as SAC. This replaces the "robustification" phase that is used in GoExplore, where the authors pull high performing trajectories to do supervised imitation learning. We argue that a policy trained with RL is better able to adapt to new situations and can better deal with distribution shift.

The random policy exploration posed a significant learning challenge, and we hypothesized that applying offline RL techniques to correct for the large percentage of the replay buffer that was collected with a different policy. We conducted some initial experiments adding a Conservative Q-learning regularizer to SAC, but we believe that further hyperparameter tuning is needed.

5 Experiments

In the following subsections, we aim to answer the questions:

1. In what problem settings does count-based exploration perform badly or inconsistently?

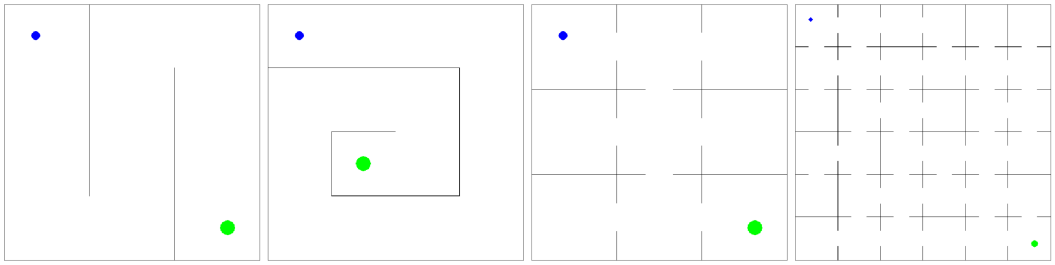

Figure 1: To test our method, we ran experiments on continuous control maze environments shown above. From left to right, we describe each maze as S-Maze, Spiral-Maze, Rooms, Large-Rooms.

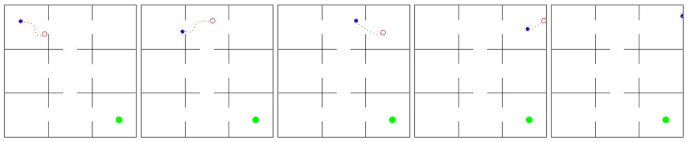

Figure 2: This is a set of frames rolling out a policy trained with a count-based exploration bonus. This run didn't converge to a policy that successfully reached the goal. Instead, it got stuck in one of the other rooms, which was a result of detachment and giving up on promising states too quickly.

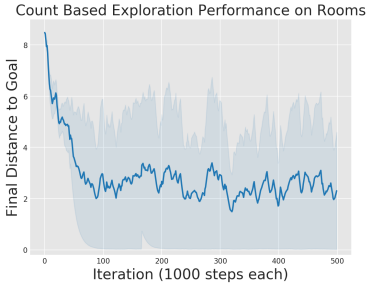

Figure 3: Here is the performance across two seeds for the Rooms task on the left, trained with a count-based bonus, which vary greatly in resulting behavior.

2. Is it practical to approximate empowerment from data and use it as a form of task-agnostic intrinsic motivation?
3. Does an online empowerment estimation scheme provide useful goals to avoid the problems seen in count-based exploration?
4. How does this scheme perform if we assume oracle resets to the desired subgoals?

5.1 Studied Environments and Count-based Exploration Shortcomings

We study the effectiveness of goal-directed exploration in four continuous control maze environments, shown in Figure 1.

Although count-based exploration, given enough iterations, can solve the S and Spiral maze environments, it isn't able to learn a good policy for more complicated environments like Rooms and Large Rooms. As shown in Figures 2 and 3, count-based exploration tends to get stuck in a room without being able to explore the final room.

One issue that count-based exploration runs into is the idea of detachment, which is described in Figure 5.

5.2 Empowerment Estimation

We follow the empowerment estimation procedure introduced in [9], with a few modifications that helped in our domains:

1. We added L2-regularization to the Gaussian channel model, which reduced incidents of wild spikes in empowerment values at certain parts of the state space that have not been explored much. This was important, because our replay buffer is not perfectly balanced in terms of its coverage of the state space, so we would often be querying uncertain empowerment values for states.

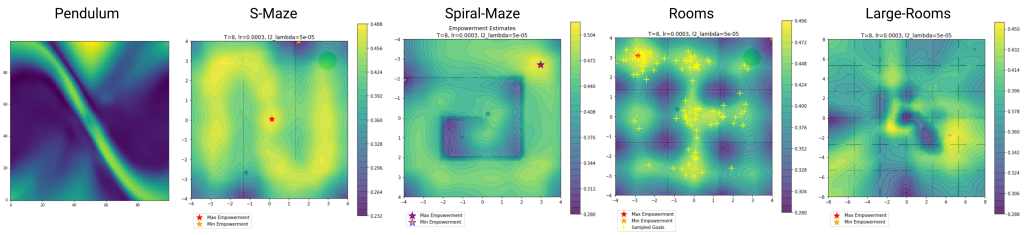

Figure 4: Offline empowerment estimates after training for 10,000 gradient steps on random trajectory data with full support of the state space. These plots are contour plots generated from empowerment estimates $\mathcal{E}(s)$ for 32×32 states s sampled in a grid within the environment boundaries.

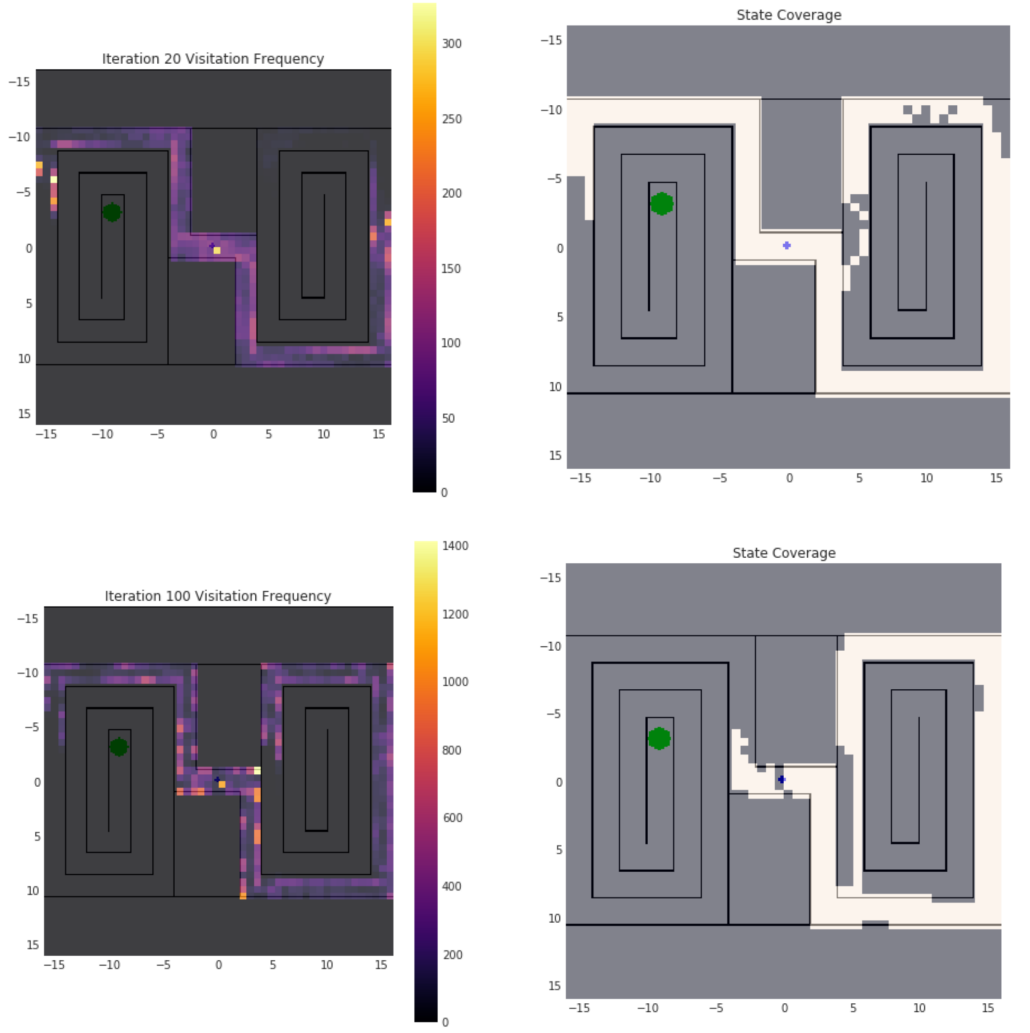

Figure 5: This figure shows an early iteration of a double sided maze task, provided a count-bonus as reward. We see that initially, the policy makes some progress toward the left side (where the goal is). However, after several iterations, the count-based bonus has been saturated in the left corridor, and the policy randomly moves on to explore the other side more. This is the problem of detachment, where a policy can forget high potential areas of the state space.

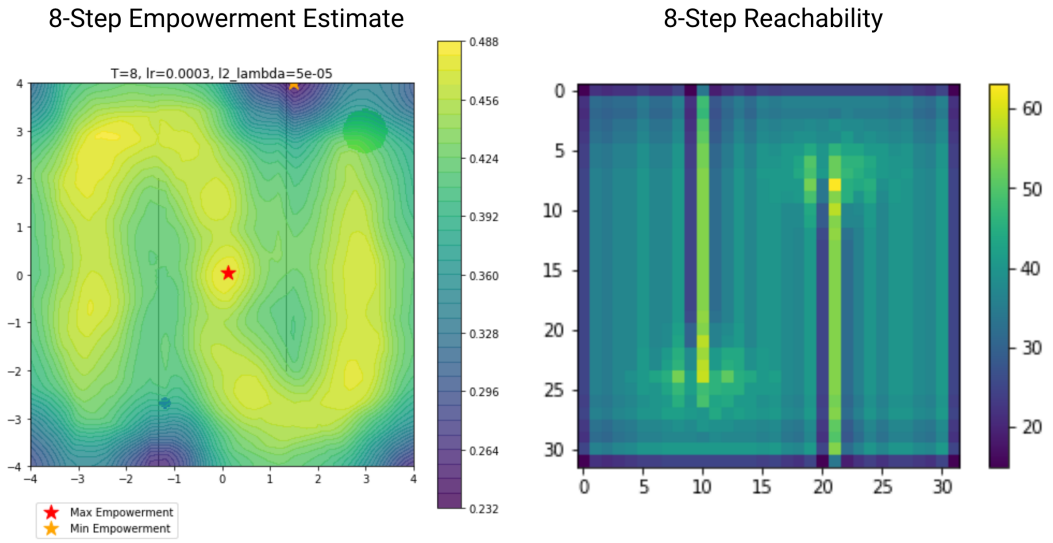

Figure 6: (left) 8-step empowerment estimates for the S-maze. (right) Empirically determined 8-step reachability, calculated from a sampled grid of states.

2. We tried a few different values for N (4, 8, 16), which is the empowerment horizon, but we settled at $N = 8$ for each of our environments, since it produced the most similar estimates to analytically calculated empowerment plots.

3. We began by training our empowerment model offline, which can be seen in figures 4, 6. We transitioned to training it online when running training it alongside the policy, which required us to cycle through the dataset in batches of size 128 in order to make sure we were optimizing the Gaussian channel model for every data point in the buffer, rather than sampling a new batch each time.

5.3 GCExplore

In this section, we compare the performance of an exploitation policy trained with data collected via a goal-conditioned navigator, choosing goals with the following heuristics:

1. Random: randomly choose states we've seen before as goals.

2. Count-based bonus: we should choose states that we haven't seen as much as goals.

3. SkewFit: we randomly sample states proportional to the inverse state density.

4. Empowerment: we should set goals at states that have high potential for exploration after reaching those states.

Figure 9 shows the performance of these different schemes on the four environments. In particular, we see that the empowerment goal generation objective performs comparably with the oracle SkewFit baseline on the S-Maze and the Rooms environment, but fails to match performance for the Sprial-Maze and Large-Room environments.

5.3.1 Unsupervised Exploration Task: Rooms Environment

As another experiment, we removed the target from the maze entirely and left the different goal schemes to do unsupervised exploration to see the expansion of state coverage over time. Figures 10 and 11 show the result of this experiment.

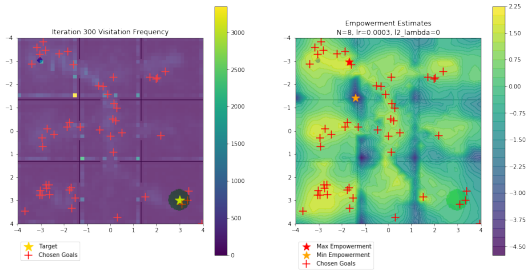

Figure 7: Online empowerment estimation on Rooms

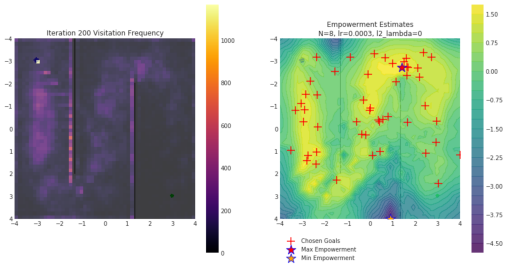

Figure 8: Online empowerment estimation on S-maze

Figure 9: We show performance of the different goal generation strategies with goal-conditioned reaching policies, followed by either random exploration or another rollout of goal-reaching. We evaluate the final distance to the goal as the metric on all four environments. The last two curves (SAC) and (SAC + Count Bonus) represents a policy that is not goal-conditioned and is just given a sparse -1/0 reward at the goal.

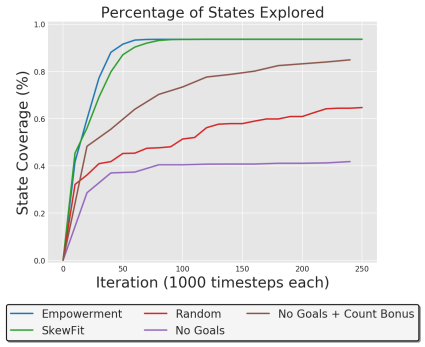

Figure 10: State coverage of unsupervised exploration an unsupervised Rooms environment at different iterations.

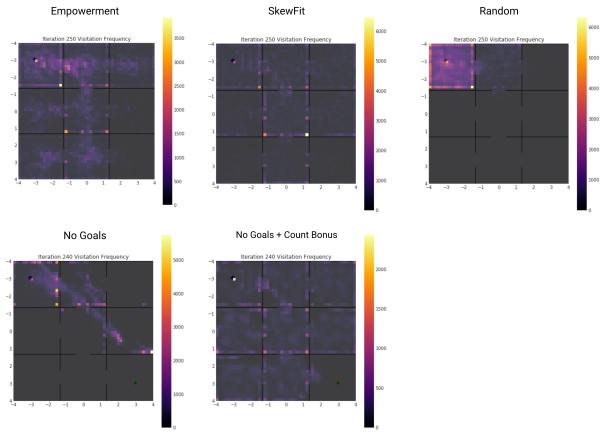

Figure 11: Iteration 250 state visitation frequency plots for each of the methods in Figure 10. Of these methods, goal-conditioned exploration with empowerment and SkewFit goal generation schemes, as well as count-based exploration, have the most spread out state visitations. Empowerment has the added benefit of exploring near critical junctions as seen in the center of each room. Random goal selection faces the problem of expanding the fringe of reachable states very slowly, and SAC with no goals gets stuck early on in training.

Figure 12: Performance of the three goal generation schemes, combined with oracle resets to those goals, and then random exploration in the same style as GoExplore.

5.4 Ablation: GoExplore Style Exploration

Lastly, we perform an ablation study of a similar procedure to GoExplore, where we assume oracle resets to the goals set by the the goal generation schemes we discussed earlier. Figure 12 shows the results of this experiment.

6 Discussion

Using a goal-conditioned policy to return to promising states is an effective way to explore an environment. Using empowerment for exploration is shown to work well for hard environments. However, we would like to note that in very hard environments such as Large Rooms, goal-conditioned exploration was unable to reach the goal. We believe this is due to difficulty for goal-conditioned RL to learn policies that reach far away goals.

7 Contributions

Justin worked on developing the goal generation interface, empowerment estimation, as well as launching experiments and helping Jonathan debug the goal-conditioned code.

Jonathan worked on implementing the base goal-conditioned exploration algorithms as well as launching experiments for these methods.

References

[1] Marcin Andrychowicz, Filip Wolski, Alex Ray, Jonas Schneider, Rachel Fong, Peter Welinder, Bob McGrew, Josh Tobin, Pieter Abbeel, and Wojciech Zaremba. Hindsight experience replay, 2017.

[2] Marc G. Bellemare, Sriram Srinivasan, Georg Ostrovski, Tom Schaul, David Saxton, and Remi Munos. Unifying count-based exploration and intrinsic motivation, 2016.

[3] Bradly C, Sergey Levine Stadie, and Pieter Abbeel. Incentivizing exploration in reinforcement learning with deep predictive models, 2015.

[4] Adrien Ecoffet, Joost Huizinga, Joel Lehman, Kenneth O. Stanley, and Jeff Clune. Go-explore: a new approach for hard-exploration problems, 2020.

[5] Tuomas Haarnoja, Aurick Zhou, Pieter Abbeel, and Sergey Levine. Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor, 2018.

[6] Shakir Mohamed and Danilo Jimenez Rezende. Variational information maximisation for intrinsically motivated reinforcement learning, 2015.

[7] Deepak Pathak, Pulkit Agrawal, Alexei A. Efros, and Trevor Darrell. Curiosity-driven exploration by self-supervised prediction, 2017.

[8] Haoran Tang, Rein Houthooft, Davis Foote, Adam Stooke, Yan Duan Xi Chen, John Schulman, Filip De Turck, and Pieter Abbeel. exploration: A study of count-based exploration for deep reinforcement learning, 2016.

[9] Ruihan Zhao, Pieter Abbeel, and Stas Tiomkin. Efficient online estimation of empowerment for reinforcement learning, 2020.