

A SOME ANALYSIS BETWEEN COUNT-BASED APPROACHES AND OUR APPROACH

Consider two corridors, left and right. The left corridor has length T_l while the right corridor has length T_r . Both starts with an initial state s_0 . When the game starts, the agent was always placed at s_0 .

For simplicity, we just assume there is only a binary action (left or right) to be chosen at the starting point s_0 , after that the agent just moves all the way to the end of the corridor and the game restarts. And we set the discount factor $\gamma = 1$.

Using count-based approach, the accumulated intrinsic reward received by the agent if moving to the left corridor for the i -th time is $R_l := T_l/i$. This is because for each state, the first time the agent visit it, it gets a reward of 1 and the second time it gets a reward of $1/2$, etc. And this is true for every state in this corridor. Similarly, for i -th time moving right, it is $R_r := T_r/i$.

Now let's think how the policy of the agent is trained. In general, the probability to take action left or right is proportional to the accumulated reward. In Q learning it is an exponential moving average of the past reward due to the update rule $Q(s, a) \leftarrow (1 - \alpha)Q(s, a) + \alpha R$ for α typically small (e.g., 0.01). If we think about Q learning starting with $Q(s_0, \text{left}) = Q(s_0, \text{right}) = 0$, and consider the situation when the agent has already done n_l left and n_r right actions, then we have:

$$Q(s_0, \text{left}) = T_l \alpha \sum_{i=1}^{n_l} \frac{(1 - \alpha)^{n_l - i}}{i} \quad (4)$$

Similarly, we have

$$Q(s_0, \text{right}) = T_r \alpha \sum_{i=1}^{n_r} \frac{(1 - \alpha)^{n_r - i}}{i} \quad (5)$$

We could define $\phi(n) := \alpha \sum_{i=1}^n \frac{(1 - \alpha)^{n - i}}{i}$ and thus $Q(s_0, \text{left}) = T_l \phi(n_l)$ and $Q(s_0, \text{right}) = T_r \phi(n_r)$.

If we treat $\phi(\cdot)$ as a function in the real domain, and assign visiting probability $\pi(a|s_0) \propto Q(s_0, a)$, then the visitation counts, now is denoted as x_l and x_r since they are continuous, for left and right satisfies the following differential equations:

$$\dot{x}_l = \frac{T_l \phi(x_l)}{T_l \phi(x_l) + T_r \phi(x_r)}, \quad \dot{x}_r = \frac{T_r \phi(x_r)}{T_l \phi(x_l) + T_r \phi(x_r)} \quad (6)$$

In the following, we will show that if $T_l \neq T_r$, long corridor will dominate the exploration since it has a lot of rewards until very late. Suppose $x_l(0) = x_r(0) = 1$ and $T_l > T_r$, that is, left side has longer corridor than right side. Then from the equations, very quickly $x_l > x_r$. For α close to 0, we could check that $\phi(x)$ is a monotonously increasing function when x is a small positive number, since the more an agent visits a corridor, the more reward it obtains. So this creates a positive feedback and the growth of x_l dominates, which means that the policy will almost always visit left. This happens until after x_l is large enough and $\phi(x_l)$ starts to decay because of diminishing new rewards and the training discount α . The decay is on the order of like $\frac{1}{x_l}$, then the agent will finally start to visit the right side. Indeed, $\pi(\text{right}|s_0) > 50\%$ if $T_r \alpha > T_l/x_l$, or $x_l > \frac{T_l}{T_r} \frac{1}{\alpha}$. In comparison, if we pick sides in a uniform random manner, when x_l is this big, x_r should also be comparable (while here x_r can be as small as 1 or 2).

If an agent have K corridors to pick with $T_1 > T_2 > \dots > T_K$, things will be similar.

When $T_l = T_r$, the dynamics is perfectly symmetric and the agent visits both left and right with $1/2$ probability. Since this symmetry is quite rare, count-based exploration is often biased towards one or two corridors (See Tbl. 1), BeBold uses difference of inverse visitation counts as intrinsic rewards, and thus balance the exploration.

B HYPERPARAMETERS FOR MINIGRID

Following (Campero et al., 2020), we use the same hyperparameters for all the baselines. For ICM, RND, IMPALA, RIDE and BeBold, we use the learning rate 10^{-4} , batch size 32, unroll



Figure 8: On policy state density heatmaps $\rho_\pi(s_t)$. BeBold continuously pushes the frontier of exploration from Room1 to Room7.

length 100, RMSProp optimizer with $\epsilon = 0.01$ and momentum 0. We also sweep the hyperparameters for BeBold: entropy coefficient $\in \{0.0001, 0.0005, 0.001\}$ and intrinsic reward coefficient $\in \{0.01, 0.05, 0.1\}$. We list the best hyperparameters for each method below.

BeBold. For all the Obstructed Maze series environments, we use the entropy coefficient of 0.0005 and the intrinsic reward coefficient of 0.05. For all the other environments, we use the entropy coefficient of 0.0005 and the intrinsic reward coefficient of 0.1. For the hash table used in ERIR, we take the raw inputs and directly use that as the key for visitation counts.

AMiGo. As mentioned in (Campero et al., 2020), we use batch size of 8 for student agent and batch size of 150 for teacher agent. For learning rate, we use learning rate of 0.001 for student agent and learning rate of 0.001 for teacher agent. We use an unroll length of 100, entropy cost of 0.0005 for student agent and entropy cost of 0.01 for teacher agent. Lastly we use $\alpha = 0.7$ and $\beta = 0.3$ for defining IRs in AMiGo.

RIDE. Following (Raileanu and Rocktäschel, 2020), we use entropy coefficient of 0.0005 and intrinsic reward coefficient of 0.1 for key corridor series of environments. For all other environments, we use entropy coefficient of 0.001 and intrinsic reward coefficient of 0.5.

RND. Following (Campero et al., 2020), we use entropy coefficient of 0.0005 and intrinsic reward coefficient of 0.1 for all the environments.

ICM. Following (Campero et al., 2020), we use entropy coefficient of 0.0005 and intrinsic reward coefficient of 0.1 for all the environments.

IMPALA. We use the hyperparameters introduced in first paragraph of this section for the baseline.

C ANALYSIS FOR MINIGRID

In addition to the analysis provided before, we also conduct some other analysis of BeBold in MiniGrid.

On Policy State Density in MiniGrid We also plot the on policy state density $\rho_\pi(s)$ for different checkpoint of BeBold. We ran the policy for 2K steps and plot the BeBold IR based on the consecutive states in the trajectory. In Fig. 8, we can clearly see that the boundary of explored region is moving forward from Room1 to Room7. It is also worth noting that although the policy focuses on exploring one room (one major direction to the boundary.) at a time, it also put a reasonable amount of effort visiting the previous room (other directions of to the boundary). Thus, BeBold greatly alleviate the short-sighted problem aforementioned.

D HYPERPARAMETER FOR NETHACK

For general hyperparameters, we use optimizer RMSProp with a learning rate of 0.0002. No momentum is used and we use $\epsilon = 0.000001$. The entropy cost is set to 0.0001. For RND and BeBold, we scale the the forward distillation loss by a factor of 0.01 to slow down training. We adopt the intrinsic reward coefficient of 100. For the hash table used in ERIR, we take several related infor-

mation (e.g., the position of the agent and the level the agent is in) provided by (Küttler et al., 2020) and use that as the key for visitation counts.