Flipping Coins to Estimate Pseudocounts for Exploration in Reinforcement Learning

Sam Lobel [*]
Department of Computer Science
Brown University
Providence, RI 02906
samuel_lobel@brown.edu

Akhil Bagaria [*]
Department of Computer Science
Brown University
Providence, RI 02906
akhil_bagaria@brown.edu

George Konidaris
Department of Computer Science
Brown University
Providence, RI 02906
gdk@cs.brown.edu

Abstract

Count-based exploration can lead to optimal reinforcement learning in small tabular domains. But, it is challenging to keep track of visitation counts in environments with large state spaces. Previous work in this area has converted the problem of learning visitation counts to that of learning a restrictive form of a density model over the state-space. Rather than optimizing a surrogate objective, our proposed algorithm *directly* regresses to a state's visitation count. Compared to previous work, we show that our method is significantly more effective at deducing ground truth visitation frequencies; when used as an exploration bonus for a model-free reinforcement learning algorithm, our method outperforms existing approaches.

Keywords: Exploration, Reinforcement learning

[*]Equal Contributors

1 Introduction

Effective exploration is key to solving long-horizon problems using reinforcement learning. When the number of possible states is small, an agent can keep track of how many times it has visited each state. This count can then be used as an exploration bonus to train an optimal policy [Strehl and Littman, 2008]. When the environment has a large number of states, the agent may not visit the same state twice. To facilitate count-based exploration in such domains, the notion of visitation counts are generalized to that of "pseudocounts". Previous methods have equated the problem of estimating pseudocounts to the canonical machine learning problem of *density estimation*: the lower the probability density (under the learned model) of a given state, the higher the reward for reaching it [Bellemare et al., 2016, Ostrovski et al., 2017].

While providing the first way to estimate pseudocounts, Bellemare et al. [2016]'s relationship between counts and probability densities exists only when the density model meets the following restrictions [Ostrovski et al., 2017]:

- it must allow computing normalized probability densities, which precludes many powerful density models
- it must be *learning-positive*, which means that the probability density of a state must increase when it is encountered by the density model again,
- it must be updated on a state exactly once per visitation, precluding common techniques such as batching and replay.

Rather than designing a density model under such a restrictive setting, we propose to directly learn the exploration bonus associated with different states. Our proposed method does not place any restrictions on the type of function approximator used and can be trained using any and all of the common deep learning practices (such as using optimizers with adaptive learning rates, batching, experience replay and so on) [Goodfellow et al., 2016].

Given all the restrictions imposed by Bellemare et al. [2016]'s pseudocounts, it is tempting to forego count-based exploration in favor of other novelty estimates that have fewer theoretical guarantees. Random Network Distillation (RND) [Burda et al., 2019] is one such popular method that has achieved state-of-the-art performance on some hard exploration problems in the Arcade Learning Environment (ALE). RND assigns an exploration bonus to a state using a simple, elegant heuristic: the novelty of a state is directly proportional to how the accurately a learned network can mimic a randomly-initialized network's projection. If s is a state, f_θ is a neural network being learned and f_{θ^*} is a fixed randomly initialized neural network, the RND exploration bonus is given by $r_i(s) = ||f_\theta(s) - f_{\theta^*}(s)||^2$.

The output of a random network f_{θ^*} can have different scales depending on the random initialization of θ^* or the input state s. This makes RND sensitive to hyperparameter settings. Furthermore, unlike visitation counts, RND's exploration bonus does not have an intuitive interpretation—it is an unnormalized distance in a neural network's latent space. Normalization tricks are often used to stabilize training in practice, but they introduce their own unintended artifacts.

Another shortcoming of RND is that it conflates *encountering* a state and *updating* a model on that state. An RL agent may encounter a state exactly once, but the novelty associated with that state will fall every time RND's prediction model f_θ is updated on it. This is not ideal because a single gradient update is often insufficient for learning effective representations in high-dimensional spaces. As we will see in the next section, our procedure gets counts from the fixed point of an optimization procedure, allowing us to train our model to convergence.

Our core insight is that a state's visitation count can be derived from the sampling distribution of Bernouli trials made every time a state is encountered. When we use a neural network to predict this sampling distribution, we inadvertently learn the inverse of the state's visitation count. We test this procedure on a visual grid world and show that our method can recover the ground truth counts while other pseudocount methods cannot. When used as an exploration bonus, it causes a model-free RL agent to rapidly explore its environment and outperform strong model-free baselines.

2 Coin Flip Network (CFN)

Consider the fair coin flip distribution of -1 and +1. Imagine you flip this coin n times, and average the results into y_n. In expectation the average is 0, but any given time you run this experiment you likely get a non-zero value for y_n. For example, if you flip the coin once you will get $Pr(y_1 = -1) = 0.5$, and $Pr(y_1 = 1) = 0.5$. If you flip it twice, you get $Pr(y_2 = -1) = 0.25$ and $Pr(y_2 = 0) = 0.5$ and $Pr(y_2 = 1) = 0.25$. More generally, for all n the second moment of y_n is related to the inverse-count:

$$\mathcal{M}_2(y_n) = E[y_n^2] = \sum_i Pr(y_n = i) * i^2 = 1/n. \tag{1}$$

2.1 Bonuses from states

We construct a dataset of m encountered states $\{s_1, ..., s_m\}$, and pair each state with a random vector of d coin flips $c_i \sim \{-1, 1\}^d$. The mean of n random coin flip vectors is a vector of d samples from y_n:

$$\Big(\frac{1}{n}\sum_{i=1}^{n} c_i\Big)_j \sim y_n \tag{2}$$

Consider the class of functions $\mathcal{F}$ such that $f \in \mathcal{F} : R^{|S|} \to R^d$. Our goal is to find $f^* \in \mathcal{F}$ that best predicts each c_i:

$$f^*(s) = \arg\min_f \sum_{i=1}^{m} \|c_i - f(s_i)\|^2 = \arg\min_f \sum_{i=1}^{m}\sum_{j=1}^{d} (c_{ij} - f(s_i)_j)^2 \tag{3}$$

If each state s_i is encountered exactly once, the optimization problem above will simply learn the mapping from states to their associated random vector. However we are interested in cases where there are multiple instances of some states. In that case, the best you can do is to learn the *mean* random vector for all instances of a given state. Let's look at one such state s, of which there are n occurrences, and each c_i is a d-dimensional sample from the coinflip distribution:

$$\begin{aligned}
f^*(s) &= \frac{1}{n}\sum_{i=1}^{n} c_i \\
\implies E\Big[\frac{1}{d}\|f^*(s)\|^2\Big] &= \frac{1}{d}\sum_{j=1}^{d} E\Big[\Big(\sum_{i=1}^{n}\frac{c_{ij}}{n}\Big)^2\Big] && \text{[Taking expectation of the vector norm]} \\
&= \frac{1}{d}\sum_{j=1}^{d} E\Big[y_n^2\Big] && \text{[Using Eq 2]} \\
&= \frac{1}{d}\sum_{j=1}^{d}\frac{1}{n} = \frac{1}{n} && \text{[Using Eq 1]}
\end{aligned}$$

This suggests that in expectation, $f^*(s)$ contains an unbiased and consistent estimator of the inverse-count. **We can now use this property of f^* to map states to bonuses.** We train a neural network $f_\theta(s)$ to approximately solve the optimization problem described in Equation 3, and obtain exploration bonuses using:

$$\mathcal{B}(s) := \sqrt{\frac{1}{d}\|f_\theta(s)\|^2} \approx \frac{1}{\sqrt{N(s)}} \tag{4}$$

2.2 Bonuses from many states

$f_\theta(s)$ is a deterministic function that maps each input to a single output: that's why the best we can do is learn the *mean* of a state's random vectors. But, we can learn a different mean for each state. A learning architecture with infinite capacity would learn the exact mean for each unique state. However, finite capacity and training time imply that the network will not learn this mapping exactly. Next, we will discuss ways in which we can control how our network trades off prediction errors among states in the data set and generalizes to unseen states during inference.

2.3 Prioritizing Novel States

Training f_θ to convergence at every time step is not feasible. To maintain reasonable training time, we update it once every time step on a single mini-batch of states drawn from its replay buffer. Revisiting the optimization target from Equation 3, we note that a *unique* state s visited n times will appear in uniform sampling n times more than a state visited only once. This would make f_θ disproportionately better at predicting the bonus corresponding to high-count states. To remedy this problem, we assign more weight to low-count states. Note that we can re-weight the optimization target from Equation 3 by scaling each state's contribution to the loss by its *true* count, without changing the ideal fixed point $f^*(s)$:

$$f^*_{\text{prioritized}}(s) = \arg\min_f \sum_{i=1}^{m} \frac{1}{\text{True Count}(s_i)}\|c_i - f(s_i)\|^2.$$

Of course, we do not have access to the true count during training; so, we approximate this procedure by prioritizing by our current *estimate* of inverse-count:

$$\text{priority}(s) \leftarrow \frac{1}{d}\|f_\theta(s)\|^2.$$

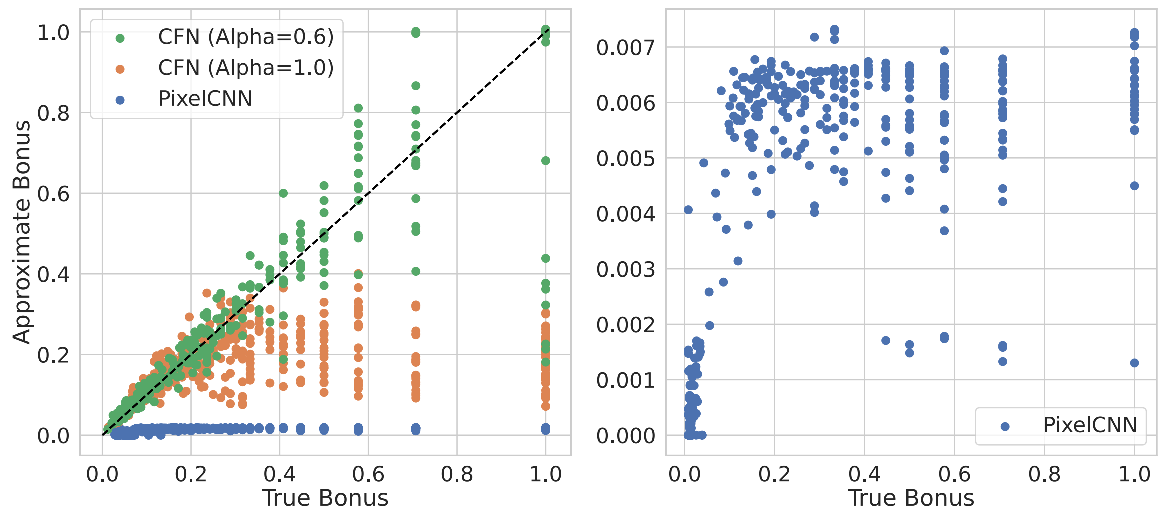

Figure 1: *(left)* Comparing bonus estimation accuracy for different variants of CFN and PixelCNN on the Visual Gridworld task (end of training). *(right)* Zoomed in version of PixelCNN's bonus prediction accuracy from the left subplot.

Prioritizing in this way introduces another difficulty: if a state has been recently added to the replay buffer, it has not appeared in many gradient updates and thus we cannot trust our estimate of its count. To combat this we also prioritize sampling by the number of times, $n_{\text{updates}}(s)$, we have sampled s in the past. We combine both these prioritization schemes using an α-weighted sum:

$$\text{priority}(s) = \alpha \left(\frac{1}{n_{\text{updates}}(s)} \right) + (1 - \alpha)\frac{1}{d}\|f_\theta(s)\|^2.$$

2.4 Temporal Consistency for Better Generalization

The training scheme described so far ignores temporal structure present in sequential decision problems; to encourage meaningful generalization, we want our training scheme to reflect the temporal relations between states. We encourage sequential states to have similar latent representations by only "replacing" each random coin flip with a probability p_r:

$$c_{i+1,j} = \begin{cases} \{-1, 1\} & \text{with probability } p_n \\ c_{i,j} & \text{with probability } (1 - p_n) \end{cases}$$

This makes the targets for sequential states similar, which in turn encourages them to have similar latent representations.

3 Experiments

We test our algorithm on a "visual grid world" task [Allen et al., 2021]. In this task, the agent gets an 84×84 image observation of the grid. The controllable block starts every episode at the bottom-left cell; it gets a terminating reward of 1 when it reaches the goal at the top-right of the grid and a reward of 0 everywhere else. For all experiments we use Rainbow DQN [Hessel et al., 2018] as a base agent and modify it with various exploration bonuses.

3.1 Evaluating Bonus Prediction Accuracy

We test CFN's ability to correctly predict the inverse visitation count in a 21×21 visual grid world. The learning agent gets image-based observations as input, but we keep track of the ground truth state (the agent's x, y location in the grid) to evaluate how well different methods can reconstruct the true bonus (or equivalently, $1/\sqrt{n_s}$). Figure 1 *(left)* shows the importance of α (as described in section 2.3)—when the ground-truth bonus is high (i.e, when the visitation count is low), CFN with $\alpha < 1$ outperforms CFN with $\alpha = 1$ because $\alpha < 1$ up-weights prediction errors on novel states. Figure 1 also shows that PixelCNN is unable to learn a meaningful count—not only are its predictions very close to 0, it assigns very similar exploration bonus to states that have been seen 25 times (which should have a true bonus of 0.2) and states that have been seen only once (which should have a true bonus of 1.0). In other words, PixelCNN does not recover the true counts in this domain *and* it dramatically underestimates the exploration bonus for truly novel states.

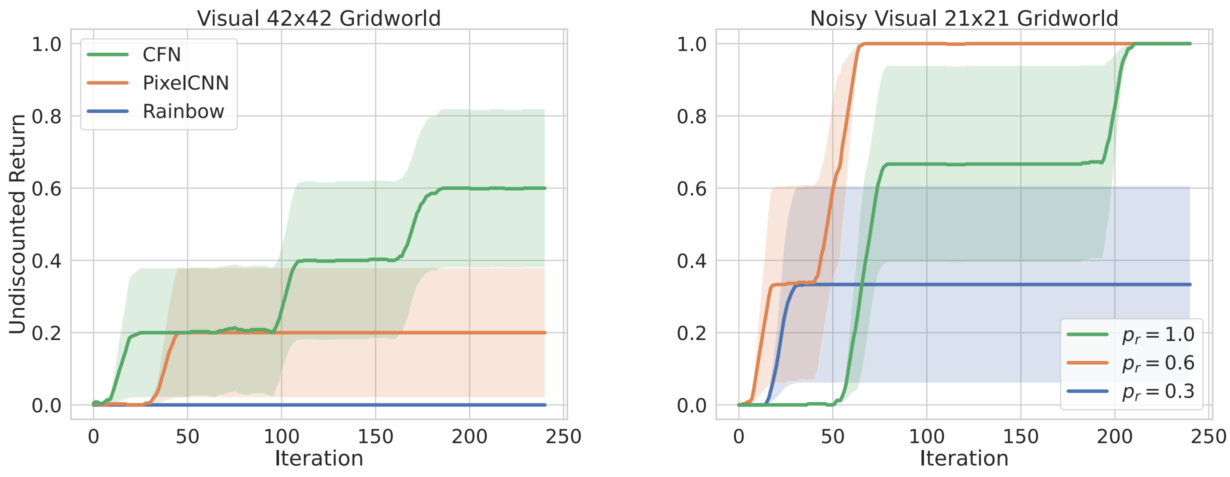

Figure 2: *(left)* Learning curves comparing CFN with Rainbow and PixelCNN on a 42×42 grid. *(right)* Comparing CFN with different values of p_r on 21×21 grid with noisy observations. Each line denotes the mean undiscounted return over 5 random seeds, shaded regions denote standard error. Each iteration is 1000 action executions in the environment.

3.2 Reinforcement Learning with CFN Bonus

RL in Visual Gridworld. We test if CFN's exploration bonus can lead to sample-efficient reinforcement learning. For this experiment, we consider a 42×42 visual gridworld. As shown in Figure 2 *(left)*, a Rainbow agent [Hessel et al., 2018] is unable to explore this grid and get any positive reward. When augmented with the CFN exploration bonus, the same agent rapidly explores the state-space and gets high return. We also compare our CFN agent to the most popular way of computing pseudocounts, PixelCNN [Ostrovski et al., 2017]; we find that CFN comfortably outperforms PixelCNN.

RL in Noisy Visual Gridworld. We now test if CFN is robust to observation noise. For this experiment, we consider a 21×21 visual grid world and add speckled noise to the visual observations. Figure 2 *(right)* shows that p_r (described in Section 2.4) helps CFN generalize to similar inputs and be robust to observation noise.

4 Conclusion

Though pseudocount-based exploration methods are a principled way to perform thorough exploration in sparse-reward reinforcement learning problems, they have not been the dominant algorithm used in practice; we aim to remedy that. We point to restrictions on the functional form, difficulty of implementation, and inability to use with a wealth of deep-learning training tricks, as a few reasons for this theory-practice divide. In contrast, CFNs estimate counts through a simple optimization procedure over the data set of encountered states, and can be used with most standard deep learning architectures. We find that CFNs explore more rapidly and represent ground-truth counts better than existing pseudocount methods. In this paper we focus on a simple environment where we can directly compute the counting accuracy; in future work, we plan to scale to more complicated domains like the hard exploration games in the ALE.

References

Cameron Allen, Neev Parikh, Omer Gottesman, and George Konidaris. Learning markov state abstractions for deep reinforcement learning. *Advances in Neural Information Processing Systems*, 34, 2021.

Marc Bellemare, Sriram Srinivasan, Georg Ostrovski, Tom Schaul, David Saxton, and Remi Munos. Unifying count-based exploration and intrinsic motivation. In *Advances in Neural Information Processing Systems*, pages 1471–1479, 2016.

Yuri Burda, Harrison Edwards, Amos Storkey, and Oleg Klimov. Exploration by random network distillation. In *International Conference on Learning Representations*, 2019. URL `https://openreview.net/forum?id=H1lJJnR5Ym`.

Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep learning*. MIT press, 2016.

Matteo Hessel, Joseph Modayil, Hado Van Hasselt, Tom Schaul, Georg Ostrovski, Will Dabney, Dan Horgan, Bilal Piot, Mohammad Azar, and David Silver. Rainbow: Combining improvements in deep reinforcement learning. In *Thirty-second AAAI conference on artificial intelligence*, 2018.

Georg Ostrovski, Marc G Bellemare, Aäron Oord, and Rémi Munos. Count-based exploration with neural density models. In *International conference on machine learning*, pages 2721–2730. PMLR, 2017.

Alexander L Strehl and Michael L Littman. An analysis of model-based interval estimation for markov decision processes. *Journal of Computer and System Sciences*, 74(8):1309–1331, 2008.