Summary of Count-Based Exploration for DRL

Ashutosh Kumar Jha, ME14B148
Mechanical Engineering, IIT Madras

***Abstract*—In this technical report, we summarize the contents and ideas presented in the technical paper titled *#Exploration: A Study of Count-Based Exploration for Deep Reinforcement Learning*[1]. We also present the possible shortcomings of the work.**

***Keywords*—Deep Reinforcement Learning, Exploration-Exploitation Dilemma, Count-Based Exploration**

I. Summary

Deep RL algorithms use Convolutional neural networks with classic RL algorithms like Actor-Critic methods [3] and Q-Learning [2] to learn good control policies in high dimensional state spaces. However, all these methods and their subsequent variants use naive exploration strategies such as ϵ-greedy algorithms. Such exploration strategies are seen to be not very effective in many environments and thus exploration strategies are an active research area in Deep Reinforcement Learning.
To summarize, the contributions of the paper can be broken down into 3 subsections:

A. Scalability for Count-Based Exploration

In the classic RL literature, another kind of exploration strategy known as **Count-Based Exploration** has been studied where a count of the number times a state is visited is stored in a tabular fashion. The agent is then given a pseudo reward in addition to the original reward which is inversely proportional to the count of the state transitioned into upon taking an action. If properly tuned, this pseudo reward will motivate the agent to take actions which will take it to those parts of the state space which are not visited frequently thus enhancing exploration.
In case of large state spaces, applying such an exploration strategy has the obvious problem of scalability where the count information can't be stored in a tabular fashion. To take care of this problem, the authors discretize the state space with the help of a hashing function(call it $\phi : \mathcal{S} \to \mathbb{Z}$). The counts can then be stored for $\phi(s)$.

B. Learning the Hashing Function

The performance of the setup will now depend strongly on the hashing function. The paper uses a particular kind of hashing called **Locality Sensitivity Hashing** where the hashing function ensures that similar items, based on a given similarity metric are hashed to the same bucket. The authors provide two different approaches to choosing the hashing function: Static Hashing and Learned Hashing.
The difference in the two approaches(at least as seen by their experiments) is primarily in terms of the kind of preprocessing function used on the state before hashing is done. The authors use a well known LSH algorithm called SimHash for both the approaches. The SimHash function is defined as:

$$\phi(s) = \text{sgn}(Ag(s)) \in \{-1, 1\}^k \tag{1}$$

where $g : \mathcal{S} \to D$ is the preprocessing function and A is a $k \times D$ matrix with i.i.d. entries drawn from a $\mathcal{N}(0, 1)$.

In static hashing, g is a predefined function (either the identity function: raw pixels or BASS function).
In Learned Hashing, the authors train an auto-encoder on the state space(in this case raw pixels). A code is extracted by down-sampling from one of the layers of the AE. The authors claim this down-sampling helps to stabilize the nature of the codes. The loss function for the auto-encoder is:

$$L\left(\{s_n\}_{n=1}^N\right) = -\frac{1}{N}\sum_{n=1}^{N}\left[\log p(s_n) - \frac{\lambda}{k}\sum_{i=1}^{D}\min\left\{(1-b_i(s_n))^2, b_i(s_n)^2\right\}\right] \tag{2}$$

As seen in the equation, the min function ensures that the codes learned are not fractional.

C. Putting things together: Experiments and Results

Once a hashing function is known, a table of counts for different "hashed" states can be maintained. A psuedo reward of the form: $\frac{\beta}{\sqrt{n(\phi(s)}}$ can be added to the original reward and any Deep RL method can be used to train the agent. The paper uses TRPO algorithm for their experiments. The paper shows results on ALE and other RLLab environments. The experiments show that the learned hashing outperforms other methods(including Vanilla TRPO) in most of the experiments. A particularly interesting study is done on the game Montezuma's Revenge where information apart from the raw-pixel information is given to the agent to achieve state-of the-art performance. In the study, it is seen that when the state space is binned into cells, performance improves greatly. I personally feel that this is because such binning implies that the "different" states would now become those in which the character literally (more like visually) goes into a different part of the room.

II. Shortcomings

In this section we make a note of possible short comings of this paper:

- The learned hashing section is counter-intuitive in the sense that when the AE just starts learning, the hashing function will be bad and thus it may lead to very bad results. The authors don't give any explanation to this.
- There is no explanation as to why LSH is used. That is why do we want similar states to be merged. Who decides what kind of "similar" is afterall good for exploration. While not necessary, the AE depends on raw pixels and is thus mostly learning that visually

similar spaces are ”similar”. I’m guessing this is the case because when binning in terms of visual pixels was done in Montezuma’s revenge, it enhanced exploration.

- The paper claims that it is indeed state-of-the-art on Montezuma’s Revenge. I personally feel that this is not a fair comparison because they didn’t compare with other DRL algos when the same extra information was provided to them.

III. Conclusion

The paper introduces a new way of exploring in the DRL domain. Its primary contributions are in terms of showing how to discretize state spaces in a high-dimensional environment and introducing count-based exploration to the DRL setting. While the results obtained by the paper are impressive, I would personally like to see more analysis of the results. A few analyses are provided in the appendix based on the kinds of codes learned but no concrete conclusions are made from these analysis.

References

[1] #Exploration: A Study of Count-Based Exploration for Deep Reinforcement Learning, Haoran Tang, Rein Houthooft, Davis Foote, Adam Stooke, Xi Chen, Yan Duan, John Schulman, Filip De Turck, Pieter Abbeel. In Neural Information Processing Systems (NIPS), Long Beach, CA, December 2017.

[2] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A. Rusu, Joel Veness, Marc G. Bellemare, Alex Graves, Martin Riedmiller, Andreas K. Fidjeland, Georg Ostrovski, Stig Petersen, Charles Beattie, Amir Sadik, Ioannis Antonoglou, Helen King, Dharshan Kumaran, Daan Wierstra, Shane Legg, Demis Hassabis. ”Human-level control through deep reinforcement learning”, in Nature 518, 529533 , 2015.

[3] Volodymyr Mnih, Adri Puigdomnech Badia, Mehdi Mirza, Alex Graves, Timothy P. Lillicrap, Tim Harley, David Silver, Koray Kavukcuoglu.”Asynchronous Methods for Deep Reinforcement Learning”, in ICML, 2016.