Multi-Step Prediction for Curiosity Driven Exploration

Kushantha U. Attanayake
kushan@umich.edu
University of Michigan

Ruchir Aggarwal
aggarwr@umich.edu
University of Michigan

Dennis Li
dennli@umich.edu
University of Michigan

Julio Soldevilla
jsolde@umich.edu
University of Michigan

Poorani Ravindhiran
poorani@umich.edu
University of Michigan

Abstract

Reinforcement learning has made great strides at exploring virtual environments, especially those of video games because of their well structured reward systems. However, it struggles in sparse reward scenarios. Curiosity driven exploration [3], where the agent is intrinsically motivated to explore the environment, has been proposed as a solution to this issue. However, current implementations of curiosity based learning only predict one time step into the future. As a result, if a catastrophic event were to happen two time steps away, the agent would be none the wiser. We propose an improvement to the proposed algorithm my generating predictions multiple steps into the future, and by using all predictions in our definition of curiosity.

1 Problem to solve

Several reinforcement algorithms aim to learn policies used to achieve tasks by maximizing a reward provided by the environment. In real world scenarios, such extrinsic rewards (provided by the environment) are very sparse. It is, in this situation, that intrinsic rewards become useful to learn policies used to achieve tasks. In a series of papers [3], [1], Deepak et. al. proposed a method to generate an intrinsic reward based on the agent's curiosity (defined as the error in an agent's ability to predict consequences of its own actions). The results are very promising, but predicting only one time step into the future may be too short-sighted. Indeed, we can easily think of many scenarios in both the real world and the game world in which seeing something truly curious may necessitate boredom in the immediate future. Currently, the proposed architecture utilizes an embedding of the state (s_t) at time t into a feature space and the action (a_t) taken at time t by the agent to generate a prediction of the embedding of the state at time $t+1$ ($\hat{\phi}(s_{t+1})$) into feature space. This predicted value is used to generate the intrinsic reward of the agent, its curiosity. Some of the questions that arise from this approach are:

1. What would happen if the agent also tried to predict states $s_{t+2}, s_{t+3}, ..., s_{t+n}$ and compared these predictions to reality?
2. What would happen if we used the embedded state $\phi(s_t)$ to make the prediction of $\hat{\phi}(s_{t+1})$?

In this paper, we attempt to answer these questions by presenting modifications to the proposed architecture in [3] and we believe that this will allow the agent to explore the environment faster and make further approaches during the game, improving its performance.

23nd Conference on Neural Information Processing Systems (NeurIPS 2019), Ann Arbor, Michigan.

2 Previous Work

In formal Reinforcement Learning algorithms, an agent tries to maximize its rewards in response to its action in the environment. These rewards are usually external in nature. But this approach requires a careful designing of the reward function. Moreover, this reward function is very specific to the environment the agent interacts with and cannot be generalized to other environments. Thus, one of the current research areas in reinforcement learning *intrinsic motivation*. In intrinsic motivated reinforcement learning, the agent internally generates a small reward for every good action it takes, according to its policy, and penalizes for bad action.

Through this project, we are particularly interested in exploring the use of "curiosity" as an intrinsic reward. Babies are curious about what happens if they move their fingers in certain way, creating little experiments that leads to initially novel and surprising but eventually predictable sensory inputs. Our 'curious' agent can be thought of in a similar way where it tries to generates such little experiments (make predictions) to keep itself surprised(or curious). An agent is encouraged to explore novel states and to make actions that will eventually reduce the error in the agent's prediction of the consequence of certain actions. Many versions of reward function based on curiosity has been proposed over time. Schmidhuber did extensive work on confidence-based curiosity and the ideas of exploration and shaping rewards. In his paper [4], he defined the curiosity proportional to the difference between how many computational resources(storage space and time) the agent needs to encode the data sequence before and after learning. Still and Precup in their paper [5] used information-theory to define the reward function as the gain in the information about the environment by the agent. Satinder et al. [2] derived their inspiration for reward function from the novelty response of dopamine neurons.

In [3], Pathak et. al. defined *curiosity* as 'the error in an agent's ability to predict the consequence of its own actions in a visual feature space learned by a self-supervised inverse dynamics model'.

3 Theory behind Curiosity Exploration

In [3], Pathak et. al. devise an agent composed by a reward generator which outputs a curiosity-driven intrinsic reward and a policy that outputs a sequence of actions that maximize the reward signal. The policy is represented by a deep neural network $\pi(s_t; \theta_P)$ with parameters θ_P, where the θ_P is optimized to maximize the expected sum of rewards

$$\max_{\theta_P} \mathbb{E}_{\pi(s_t;\theta_P)}[\Sigma_t r_t] \tag{1}$$

The main contribution of the paper is the design of an architecture to compute this intrinsic reward signal based on prediction error of the agent's knowledge about its environment.

The key insight that allows for the development of a good intrinsic reward is modeling the states in a feature space that distinguishes things that can be controlled by the agent, things that the agent does not control but still affect the agent and things that the agent does not control and that do not affect the agent. To learn such a feature space, the researchers train a deep neural network with two components: the first one will encode the raw state s_t into a feature vector $\phi(s_t)$ and the second one will take as inputs the feature vectors $\phi(s_t), \phi(s_{t+1})$ and predict the action taken by the agent to go from state s_t to state s_{t+1}. This process corresponds to learning a function g, called the *inverse dynamics model*, and defined by

$$\hat{a}_t = g(s_t, s_{t+1}; \theta_I) \tag{2}$$

, where $\hat{a}_t$ is the prediction of the action a_t and the parameters θ_I are optimized to solve

$$\min_{\theta_I} L_I(\hat{a}_t, a_t)$$

where L_I is some function measuring the difference between the predicted and actual actions.

Additionally, the authors train another network that will take as inputs a_t and $\phi(s_t)$, and will learn the function f that will be used as the feature encoding of the state at the next time-step

$$\hat{\phi}(s_{t+1}) = f(\phi(s_t), a_t; \theta_f)$$

where $\hat{\phi}(s_{t+1})$ is the prediction and θ_F are parameters trained to optimize

$$L_F(\phi(s_t), \hat{\phi}(s_{t+1})) = \frac{1}{2}\|\hat{\phi}(s_{t+1}) - \phi(s_{t+1})\|_2^2$$

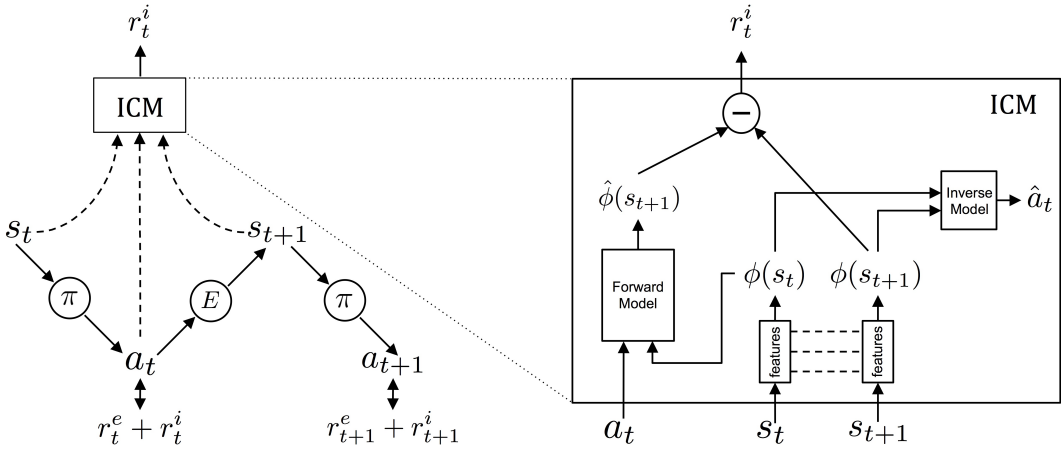


Figure 1: Original Architecture

The authors call this learned function, f, the *forward dynamics model*. With these two models defined as above, the proposed intrinsic reward function is then defined as

$$r_t^i = \frac{\eta}{2}\|\hat{\phi}(s_{t+1}) - \phi(s_{t+1})\|_2^2 \tag{3}$$

where η is some scaling factor. Finally, since we are interested in finding the optimal reward and the optimal parameters for the network g and f, the overall optimization problem is solved so that the agent learns the following

$$\min_{\theta_P,\theta_I,\theta_F} \left[-\lambda \mathbb{E}_{\pi(s_t;\theta_P)}[\Sigma_t r_t] + (1-\beta)L_I + \beta L_F\right] \tag{4}$$

In [1],the main contribution of the authors is the study of different ways of obtaining the feature space of the states of the game and understand how the agent learns in these different feature spaces. Fundamentally, they establish the baselines that any good feature space should have. As per that, any good feature space should be compact, sufficient and stable. Some of the feature space transformations discussed include mere pixel observation, random features initialization (obtained by doing a random initialization for the embedding network and fixing this initialization; no training of the network), variational auto-encoders (VAE) and inverse dynamics features (IDF- obtained from the inverse dynamics model described above).

3.1 Network details

The intrinsic curiosity module consists of the forward and the inverse model. The inverse model uses a series of four convolution layers, each with 32 filters, kernel size 3x3, stride of 2 and padding of 1. ReLU non-linearity is used after each convolution layer. The dimensionality of the output of the fourth convolution layer is 288. Two such outputs for consequent time are concatenated into a single feature vector and passed as inputs into a fully connected layer of 256 units followed by an output fully connected layer with 4 units to predict one of the four possible actions. The forward model is constructed by a sequence of two fully connected layers with 256 and 288 units respectively.

4 Multi-Step Prediction

We propose two major changes to the implementation provided in [3]. First, we modify the forward dynamics model to generate $\hat{\phi}s_{t+2}, \hat{\phi}s_{t+3}, ..., \hat{\phi}s_{t+n}$ in addition to generating $\hat{\phi}(s_{t+1})$, and we redefine the reward signal to use both of these predictions such that

$$r_t^i = \frac{\eta}{2}\sum_{i=1}^{n}(w_p^i||\hat{\phi}(s_{t+i}) - \phi(s_{t+i})||)$$

where w_p and η are scaling factors, $\hat{\phi}(\cdot)$ are the outputs generated by the forward dynamics model, and $\phi(\cdot)$ are the output generated by the feature extractor. In addition, for now, we choose to use 'Random Feature Initialization' as our feature learning method in the feature extractor. We take our embedding network (CNN) and fix it after random initialization. Since the network is fixed, the features are stable.

To accommodate this change, we also have to change the way the policy was defined. In the original implementation, the forward dynamics model needed a_t and $\phi(s_t)$ to generate $\hat{\phi}(s_{t+1})$, thus if it were to generate $\hat{\phi}(s_{t+2})$, it would need a_{t+1} and $\phi(s_{t+1})$. We can circumvent the requirement for $\phi(s_{t+1})$ by using $\hat{\phi}(s_{t+1})$, which is generated by the forward dynamics model, but the only way to acquire a_{t+1} is through the policy network, which requires s_{t+1} as an input.

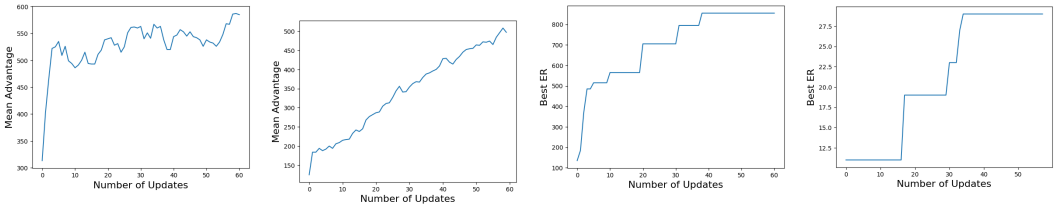
Thus, the second change we propose is to change the policy network such that the input to it would be the features of s_t, i.e. $\phi(s_t)$. This would allow us to use $\hat{\phi}(s_{t+1})$ as an alternative in the forward dynamics model, allowing us to acquire an action for a predicted state, thus allowing us to generate $\hat{\phi}(s_{t+2})$.

Our hypothesis is that the performance of the agent will improve because of the increased horizon. With the new definition of the reward, the agent is disincentivized from visiting states that are adjacent to already well known states, in addition to being disincentivized from visiting already visited states. As a result, it would even further avoid dangereous regions that could lead to it's death, and choose to explore the state space even further,

One of the major concerns was the proposed change to the policy. Because our implementation necessitates acquiring the next action using the features of the current state as the input, instead of deriving it directly from the observed state, there was concern that the agent performance would decrease, or at the very least depend heavily on the method used to extract features form the observed state. This concern has been alleviated, at least partially, which is explained in greater detail in the current progress section.

5 Results

5.1 Policy Architecture Change



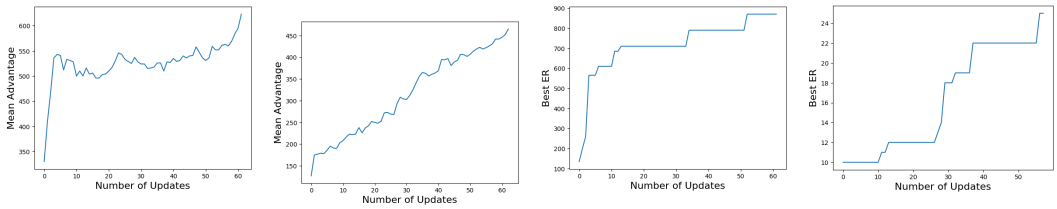
Space Invaders advantage | Breakout advantage | Space Invaders best external reward | Breakout best external reward

Results with original policy architecture

Changing the policy network to operate on the feature space of the state instead of the raw observations seemed to have a slight impact in agent performance, but the differences between the two architectures are not large enough to raise concerns.

We ran the simulation for only 63 updates which is far less than actually required (more than 1000) because of the time it takes to train the agent. From Figure 2. we can say the following:

- The average agent advantage follows similar curve as with the original architecture. We reason that this maybe because of the less number of updates and may see deviation as the number of updates increase.
- The best external return to agent are slightly different. In case of Space Invaders (column 3 in Fig. 2), the best external return is higher for our updated architecture around update



Space Invaders advantage | Breakout advantage | Space Invaders best external reward | Breakout best external reward

Results with updated policy architecture

Figure 2: Preliminary results from the initial run of the algorithm. The top row shows the results for the original architecture and the second row shows the results of the updated architecture.

40 [1]. Similarly, for Breakout (column 4 in Fig. 2), the external return is higher in updated architecture than the original architecture.

These preliminary results seem promising. Further work is required to check if the results are consistent over longer lifetimes for the agent. Furthermore, we are excited to see that even random features are sufficient for both the forward dynamics model as well as the policy network.

However we face the following issues which will play a key role in the future direction of the project:

- Training time is a big issue. The expected time to finish one full run based on default hyperparameters, with our current setup on Google Cloud, is 400+ hours [2].
- The number of environments available in gym library is big and the amount of training required is extremely high even for a single Atari game environment like Breakout. So which environments to evaluate need to be decided.

5.2 Forward Dynamics Model Change

Having changed the policy to take input from the feature/latent space rather than state space, it is now possible to generate action for the second time step, a_{t+1}, without having to wait for the second state. This, in turn, allows us to make a prediction for the feature embedding of the second state using the forward dynamics model at the first time step itself. As mentioned in section 4, our next step is to make modifications to the proposed architecture so that the intrinsic reward (curiosity) is computed as a weighted average of the state prediction errors at two time steps in advance. Mathematically, it is given by equation 5.

$$r_t^i = \frac{\eta}{2}(w_1||\hat{\phi}(s_{t+1}) - \phi(s_{t+1})|| + w_2||\hat{\phi}(s_{t+2}) - \phi(s_{t+2})||) \tag{5}$$

where w_1, w_2 and η are scaling factors, $\hat{\phi}(\cdot)$ are the outputs generated by the forward dynamics model, and $\phi(\cdot)$ is the output generated by the feature extractor.

We plan to give more weight, i.e. place more of agent's trust, on the prediction error at the following time step than at the next time step, as $\hat{\phi}(s_{t+1})$ is generated based on a_{t+1}, which in turn is generated from $\hat{\phi}(s_t)$ (not $\phi(s_{t+1})$). In other words, the prediction for the second time step is made based on the prediction at first time step, so for tasks like training the forward dynamics model, it may be wise to emphasize the first prediction. However, for the reward function, we believe it will be interesting to see how the agent behaves when reward is based entirely or almost entirely on the second prediction error. This could result in the greatest divergence in behavior from the prior model. We will maintain these weights as hyperparameters and intend to experiment with multiple settings.

This would involve modifying the network to incorporate the state's predictions and actions for two time steps as well as reformulating our loss functions (compared to [3]), incorporating both time

[1] simulated with default parameters and not with optimum parameters for the environment

[2] currently we have trained an agent for 4 hours and it still required an order of magnitude 10^2 more time to finish one run (assuming linear time)

steps for training the forward dynamics model and policy network. Once implemented, we would aim to validate our method by showing performance improvements in various game environments, especially in more complex environment like Super Mario, which we hypothesize will benefit more from looking multiple steps in the future. We believe it would valuable to discover what types of games benefit the most from looking further into the future, and if it aligns with our intuition. After we have implemented our network with random features, we may also explore the other feature learning methods like VAE, IDF et cetera.

A reach goal will be to extend our model to look k time steps ahead, where k is a hyperparameter. Our model in theory should be easy to extend to more timesteps, as we can simply feed our predicted features back into the forward model and have our loss function be the weighted sum of these k time step prediction errors. However, we are foresee a large accumulation of noise with further prediction.

If we see promising results, a further goal from this (likely not within the scope of this final project) would be to further expand upon the concept of looking into the future. This could involve making multiple distinct predictions at each timestep and thus having a branched model of the future. From there, we could combine them in some way or search for the most promising paths. In general, we find the concept of searching for an uncertain future to be exciting in both a technical and human sense, so we are excited to explore this concept and see what we may uncover.

6 Future Work

7 Division of Work

8 Conclusion

9 Acknowledgements

We would like to thank and acknowledge Ruben Villegas, who originally had the idea of extending [1] to multiple timesteps and for providing us with lots of guidance in this research. We would also like to thank Honglak Lee for his help and mentorship in this project and throughout this course.

References

[1] Yuri Burda, Harri Edwards, Deepak Pathak, Amos Storkey, Trevor Darrell, and Alexei A. Efros. Large-scale study of curiosity-driven learning. In *ICLR*, 2019.

[2] Nuttapong Chentanez, Andrew G Barto, and Satinder P Singh. Intrinsically motivated reinforcement learning. In *Advances in neural information processing systems*, pages 1281–1288, 2005.

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

[4] Jürgen Schmidhuber. A formal theory of creativity to model the creation of art. In *Computers and creativity*, pages 323–337. Springer, 2012.

[5] Susanne Still and Doina Precup. An information-theoretic approach to curiosity-driven reinforcement learning. *Theory in Biosciences*, 131(3):139–148, 2012.