Curiosity via model-disagreement driven exploration for improved Reinforcement learning

Nihaar Shah (ns3413@columbia.edu), Rob Kwiatkowski

December 23, 2020

1 Introduction

We investigate the merits of supervised pretraining using an intrinsic reward of a Reinforcement learning model and planner for efficient exploration. The objective is to achieve an improved sample complexity for the downstream Model Predictive Control task as compared to a randomly initialized model. Another goal is to achieve better performance in the downstream RL algorithms initialized with the pretrained planner when compared to a randomly initialized planner. The environment used was the Half-Cheetah v2 from the OpenAI gym for the task of locomotion of a four legged robot.

The inspiration for the model ensemble is derived from (Chua et al. 2018). The inspiration for using some form of disagreement metric as a proxy for intrinsic reward was partially derived from (Pathak, Gandhi, and Gupta 2019) and its predecessors.

2 Results

2.1 Architecture

The control flow of our training process is depicted in Figure 1. The environment and planner feed each other state and actions in a loop. After a set number of such interactions between the 'environment' and 'planner', we have collected $\{s_t, a_t, r_t, s_{t+1}\}_N$ tuples, which are stored in a replay buffer.

After this, the dynamics model ensemble is trained to predict s_{t+1} from s_t, a_t pairs sampled from this replay buffer. The loss function for the models is the Mean Squared Error.

The planner is trained at some set interval of steps. Its objective is to produce an action that maximizes the disagreement among the models' predictions of s_{t+1} in the ensemble. Formally, its objective function can be written as Equation 1. Note that the disagreement metric among the ensemble models is a function of the Model parameters as well as the planner's parameters. While doing a backward pass to update the planner weights using this disagreement

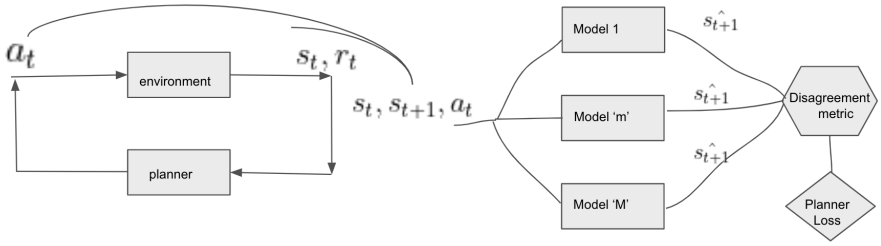


Figure 1: The flow begins with the environment which produces an state and reward. This state is fed to the planner which produces an action. After a set number of iterations in the loop, the model gets trained on a batch of states and actions sampled from the buffer. At another set interval, the planner gets trained using an objective calculated from the model predictions.

metric, we freeze the model weights to ensure that only the planner is being trained to increase the disagreement.

2.2 Outcome

Figure 2 illustrates the 4 situations we consider.

- Use Pretrained policy (pt) and load Replay buffer (bt) from the training phase. This option is denoted 'ptbt'. The buffer is loaded from experience gathered during the training and then added to during the RL phase.
- Use Pretrained policy (pt) but not Replay buffer (bf). This is denoted 'ptbf'. In this case, the buffer is empty at the beginning and populated entirely during the RL phase.
- Use random policy (pf) and load Replay buffer (bt). This is denoted 'pfbt'. The policy network is randomly initialized and entirely learned during the RL phase. The buffer is loaded at the start.
- Use random policy (pf) and do not use replay buffer (bf). Both are learner/filled during the RL phase.

3 Methodolgy

3.1 Part 1: Train the Policy Network to explore

As discussed, we are doing supervised learning of the policy using model disagreement as the objective to maximize.

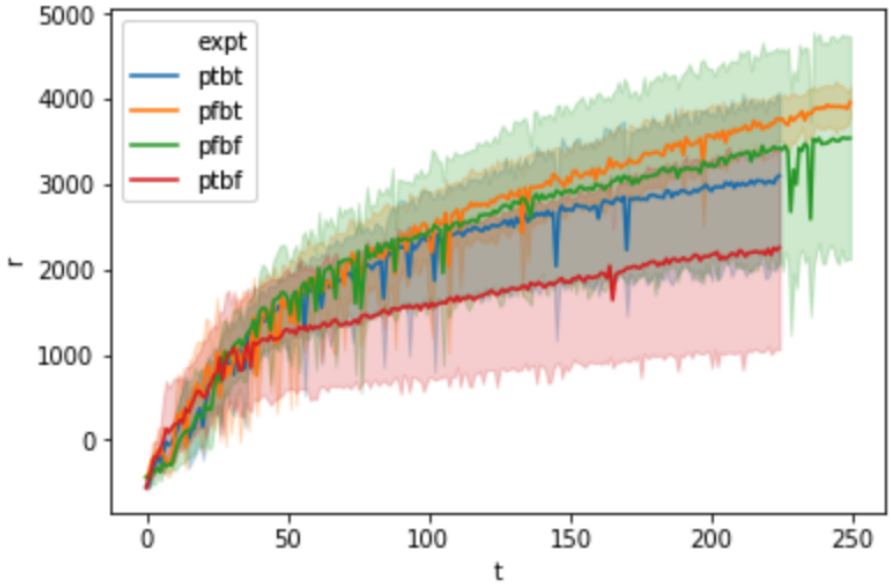
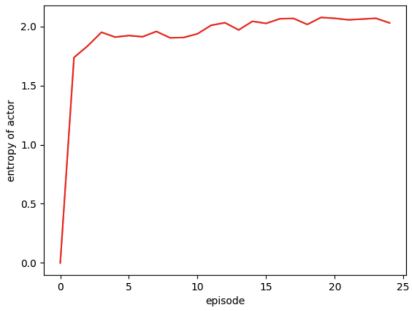
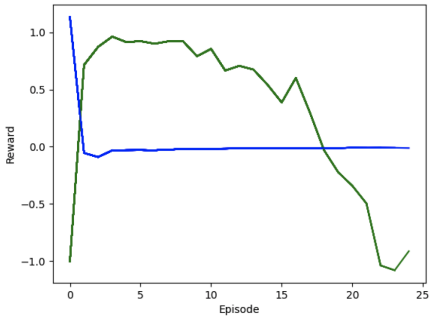


Figure 2: TD3 (Fujimoto, Hoof, and Meger 2018) algorithm extrinsic reward evaluated under the following cases: With ('pt') or without ('pf') using supervised policy pretraining and with ('bt') or without ('bf') pretrained buffer. The lighter shade around each solid line is the uncertainty band across the 5 seeds of training for each configuration.



(a) Actor entropy with episodes.



(b) Extrinsic Reward (green) and intrinsic (blue(with episodes.

Figure 3: The entropy of the planner is shown on the left. The rewards from the environment (green) and the intrinsic reward from model disagreement (blue) are displayed on the right. The intrinsic reward gently increases despite being explicitly optimized, whereas extrinsic reward falls drastically despite not being optimized as such.

$$a^* = \textbf{arg max}_{\textbf{Planner}}\ \textbf{Model-Disagreement}(s_t, s_{t+1}, a_t) \tag{1}$$

Each model is predicting the mean (μ) and variance (Σ) of the next-state distribution and the disagreement is as follows where $\forall i, j \in [0, n], \mu \in R^n$

$$\textbf{Model-Disagreement}(s_t, s_{t+1}, a_t) = (\mu[i] - \mu[j]) * e^{(\Sigma[i] - \Sigma[j])^2} \tag{2}$$

Besides monitoring the intrinsic reward as shown in Figure 4, we also plot the entropy of the planner actions and plot the extrinsic reward (which, though, is not optimized as such). 3) Val-loss on val set created using the previous round of supervised training of this policy 4)

3.2 Part 2: Modeling new states

$$s^*_{t+1} = \textbf{arg min}_{\textbf{Model}_i(s_t, a_t)}\ \textbf{MSE}(s_{t+1}, \hat{s}_{t+1}) \tag{3}$$

3.3 Visualizing and verifying training

To verify our hypothesis that learning the planner should increase the intrinsic reward, whereas learning the model should decrease it, we follow the procedure. We train both the model and policy for 5k steps.

Next, we stop training the model and continue training the policy. This curve is shown in green and can be seen to decrease after 5k steps. The y axis is the negative disagreement or intrinsic reward. The planner is trained to maximize disagreement; hence the green curve of negative disagreement dropping suggests it is indeed being trained.

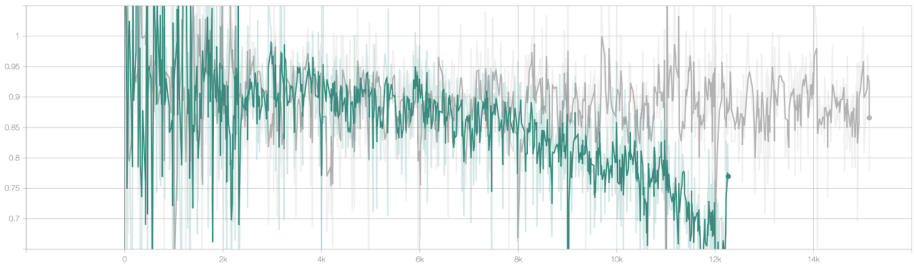


Figure 4: This test involved visualizing the training loss for the cases when 1) We stop training the planner but continue to train the model (green) 2) Stop training the model and continue to train the planner (grey)

As an alternative test, we stop training the policy but continue training the model after 5k steps. The gray line shows the intrinsic reward in that case. We would expect the disagreement to reduce (hence negative disagreement to increase) because the models repeatedly learn similar states. This is not clearly visible in the training curve, but at the same time, there seems not to be any drop in the y-axis for the gray curve either. Perhaps, this suggests a need for a more powerful signal for the disagreement metric.

4 Discussion and Future directions

The results are not as hoped, especially in terms of improving the downstream task. One indication of what is going wrong is that the intrinsic reward plot (gray) when the model is learning, and the planner is not learning shown Figure 4 doesn't increase. The gray curve not increasing might suggest the need to alter the disagreement metric to reflect that when the model is learning, there should be less disagreement. Another indication that there was a potential problem was that in Figure 3b the intrinsic reward of the planner only gently increases as the episodes of training the planner to maximize disagreement progress ahead. This also hints that the disagreement metric needs to be altered so that the reward should increase more rapidly during the training phase.

5 Code

Code for all experiments can be found at: https://github.com/rjk2147/MBExp/tree/supervised_policy_learning

References

[Chu+18] Kurtland Chua et al. "Deep Reinforcement Learning in a Handful of Trials using Probabilistic Dynamics Models". In: *CoRR* abs/1805.12114

(2018). arXiv: 1805.12114. URL: http://arxiv.org/abs/1805.12114.

[FHM18] Scott Fujimoto, Herke van Hoof, and David Meger. "Addressing Function Approximation Error in Actor-Critic Methods". In: *CoRR* abs/1802.09477 (2018). arXiv: 1802.09477. URL: http://arxiv.org/abs/1802.09477.

[PGG19] Deepak Pathak, Dhiraj Gandhi, and Abhinav Gupta. "Self-Supervised Exploration via Disagreement". In: *CoRR* abs/1906.04161 (2019). arXiv: 1906.04161. URL: http://arxiv.org/abs/1906.04161.