Paper summary: Curiosity-driven exploration by self-supervised prediction

April 5, 2022

1 Idea in few sentances

2 Explanation of the central concept

3 Methodology

4 Initial rambly notes

The paper introduces the intrinsic curiosity module (ICM). It is the paper that trains a forward prediction model and uses its loss to generate exploration reward (intrinsic reward). There are problems with it, but it is important to understand because many of the following methods try to solve the problems this approach has.

4.1 Abstract

Curiosity is formulated as the error in the agent's ability to predict the consequence of its own actions in a visual feature space learned by a self-supervised inverse dynamics model. The method ignores the aspects of the environment that cannot affect the agent, or that the agent can not affect i suppose. The paper investigates 3 scenarios:

1. sparse extrinsic reward
2. exploration with no extrinsic reward
3. generalization to unseen scenarios (or at least faster learning in new scenarios)

4.2 Introduction

Other approaches tried to do one of 2 things:

1. encourage novelty seeking (exploring unseen states)

2. encourage the agent to perform actions that reduce the error/uncertainty in the agent's ability to predict the consequence of its own actions

The problem with 1) is that a statistical model of the distribution of the environmental states is required. 2) requires a model that predicts $(\boldsymbol{a}_t|\boldsymbol{s}_t) \rightarrow (\boldsymbol{s}_{t+1})$. Both of these are hard to do in high-dimensional continuous state spaces such as images. Also the agent-environment system is inherently stochastic so the methods fail there.

4.3 Method

Reward is:

$$r_t = r_t^i + r_t^e \tag{1}$$

i.e. extrinsic + intrinsic reward, where r_t^e will mostly be 0.

The whole architecture can be seen in the following picture: In words:

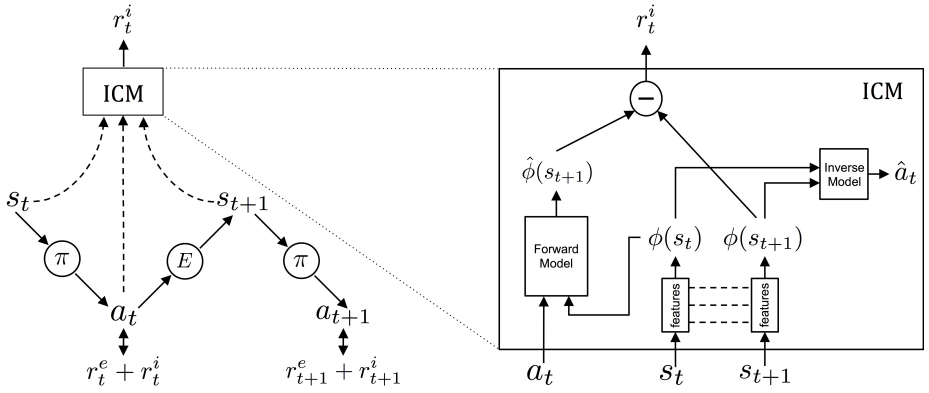

Figure 1: Architecture which includes ICM.

1: In $\boldsymbol{s}_t$ agent takes $\boldsymbol{a}_t$ sampled from π, goes to $\boldsymbol{s}_{t+1}$. π is trained to optimize $r_t = r_t^i + r_t^e$, where r_t^e comes from the environment and r_t^i comes from ICM. ICM encodes states $\boldsymbol{s}_t, \boldsymbol{s}_{t+1}$ into features $\phi(\boldsymbol{s}_t), \phi(\boldsymbol{s}_{t+1})$ that are trained to predict a_t (thus being an inverse dynamics model). The forward model takes as inputs $\phi(\boldsymbol{s}_t)$ and $\boldsymbol{a}_t$ and predicts the feature representation $\hat{\phi}(\boldsymbol{s}_{t+1})$ of $\boldsymbol{s}_{t+1}$. The prediction error in the feature space is used as the curiosity based reward signal. $\phi(\boldsymbol{s}_t)$ has no incentive to encode any environment feature that can not influence or are not influence by the agent's actions and thus the learned exploration strategy is robust to the uncontrollable aspects of the environment.

The point of the inverse dynamics model is used to learn the feature representations. The inverse model was chosen for this because it does not care about the parts of the observation which are irrelevant for predicting actions, thus resulting in a feature space constrains the observation space into the part of it relevant for the agent. And that's what we want — we don't care about backgrounds, disctracting sprites etc..

4.3.1 Self-supervised prediction for exploration

The feature space is learned by training a deep neural network with 2 sub-modules: the first sub-module encodes the raw state ($\boldsymbol{s}_t$) into a feature vector $\phi(\boldsymbol{s}_t)$ and the second submodule takes as inputs the feature encodings $\phi(\boldsymbol{s}_t), \phi(\boldsymbol{s}_{t+1})$ and predicts the action (a_t) taken when moving from $\boldsymbol{s}_t$ to $\boldsymbol{s}_{t+1}$. Training this network amount to learning a function g defined as:

$$\hat{a}_t = g\left(\boldsymbol{s}_t, \boldsymbol{s}_{t+1}; \theta_I\right) \tag{2}$$

where $\hat{a}_t$ is the predicted estimate of a_t and the neural network parameters θ_I are trained to optimize

$$\min_{\theta_I} L_I(\hat{a}_t, a_t) \tag{3}$$

If a_t is discrete, the output of g is a soft-max distribution across all possible actions and minimizing L_I amounts to maximum likelihood estimation of θ_I under a multinomial distribution. This is known as the inverse dynamics model.

In addition, another neural network is trained. It takes as inputs a_t and $\phi(\boldsymbol{s}_t)$ and predicts the feature encodign of the state at time step $t+1$:

$$\hat{\phi}(\boldsymbol{s}_{t+1}) = f\left(\phi(\boldsymbol{s}_t), a_t; \theta_F\right) \tag{4}$$

and the neural network parameters θ_F are optimized by minimizing:

$$L_F\left(\phi(\boldsymbol{s}_t), \hat{\phi}(\boldsymbol{s}_{t+1})\right) = \frac{1}{2}||\hat{\phi}(\boldsymbol{s}_{t+1}) - \phi\boldsymbol{s}_{t+1}||_2^2 \tag{5}$$

This is known as the forward dynamics model.

With this the intrinsic reward signal is:

$$r_t^i = \frac{\eta}{2}||\hat{\phi}(\boldsymbol{s}_{t+1}) - \phi\boldsymbol{s}_{t+1}||_2^2 \tag{6}$$

where η is a scaling factor.

The overall optimization problem that is solved for learning the agent is the following composition of thus far introduced equations:

$$\min_{\theta_P, \theta_I, \theta_F} \left[-\lambda E_{\pi(\boldsymbol{s}_t; \theta_P)}\left[\sum_t r_t\right] + (1-\beta)L_I + \beta L_F\right] \tag{7}$$

where $0 \leq \beta \leq 1$ is a scalar that weighs the inverse model loss against the forward model loss and $\lambda > 0$ is a scalar that weighs the importance of policy gradient loss against the importance of learning the intrinsic reward signal.

4.4 Other stuff

The analysis of the method is super interesting, but I don't feel the need to summarize it here. It was a good read, won't hurt to just read it all again later.