Automatic Goal Generation for Reinforcement Learning Agents

Alessandro Trenta - 566072

ISPR - Dip. of Informatics
Università di Pisa

26 maggio 2022

The problem

- In general RL frameworks the main objective is to find a policy $\pi(a_t|s_t)$ that maximizes the expected future return.
- In this problem, instead of maximizing the return over a single reward function we want to analyze the situation in which we have a range of reward functions r^g indexed with a goal $g \in \mathcal{G}$.
- A goal is defined as a set of states $\mathcal{S}^g \subset \mathcal{S}$. The reward function associated to this goal is

$$r^g(s_t, a_t, s_{t+1}) = \mathbb{1}\{s_{t+1} \in \mathcal{S}^g\}$$

- We will consider $S^g = \{s \in \mathcal{S} : d(f(s), g) \leq \epsilon\}$ where f projects the states in the goal space $\mathcal{G}$ and d is a distance in this space.

- Given g we consider a Markov Decision Process that terminates whenever $s_t \in \mathcal{S}^g$. We then consider the return to be $R^g = \sum_{t=0}^{T} r_t^g$. This is actually a binary random variable and indicates whether the agent is able to reach a state close enough to the goal in at maximum T time steps.
- The policy is also g dependent: $\pi(a_t|s_t, g)$ and the expected return for a goal is

$$\begin{aligned} R^g(\pi) &= \mathbb{E}_{\pi(\cdot|s_t,g)} \left[\mathbb{1}\{\exists t \in [1, \dots, T] : s_t \in \mathcal{S}^g\right] \\ &= \mathbb{P}\left(\exists t \in [1, \dots, T] : s_t \in \mathcal{S}^g\right) \end{aligned}$$

- We then assume to have a test distribution over goals p_g so that out objective is to maximize the expected mean return over goals obtaining the policy

$$\pi^*(a_t|s_t, g) = \arg\max_{\pi} \mathbb{E}_{g \sim p_g(\cdot)}[R^g(\pi)]$$

which is indeed the average probability of success over all possible goals (w.r.t. p_g).

Three main assumptions:

- A policy learned from enought goals in specific area of $\mathcal{G}$ can learn to interpolate well for other goals in this area.
- A policy trained on some set of goals is a good initialization for other goals that are close enough.
- If a goal is reachable, there is a policy that can reach it consistently.

The main catch: goals of intermediate difficulty generation

- We want to train our agent gradually. At each iteration of policy training we don't want the agent to work on goals that it can barely reach or that are too easy (other than avoiding catastrophic forgetting).
- How do we define hard and easy goals? Introduce the set of **Goals of intermediate difficulty** (for the i-th iteration):

$$GOID_i := \{g : R_{\text{min}} \leq R^g(\pi_i) \leq R_{\text{max}}\}$$

that is the goals which probability to be reached in at most T time steps with the current policy π_i is in the range $[R_{\text{min}}, R_{\text{max}}]$.

- We then want a way to generate new goals in this set in a efficient way. This is done using a Generative Adversarial Network called the goal GAN.
- To train the GAN we first give a label y_g to the goals used in the last iteration of training set to 1 if they are in $GOID_i$ and 0 otherwise. This is done by policy evaluation.

- The goal GAN is responsible for one of the key parts of the model: generate new goals of the right difficulty for current iteration. The generator G aims to generate goals g from a noise vector $z \sim p_z(\cdot)$. The discriminator has to distinguish goals in $GOID_i$ from those that are not.
- The two losses are defined as:

$$\begin{aligned} V(D) &= \mathbb{E}_{g \sim p_{\text{data}}(g)} \left[y_g (D(g) - b)^2 + (1 - y_g)(D(g) - a)^2 \right] \\ &\quad + \mathbb{E}_{z \sim p_z(z)} \left[(D(G(z)) - a)^2 \right] \\ V(G) &= \mathbb{E}_{z \sim p_z(z)} \left[(D(G(z)) - c)^2 \right] \end{aligned}$$

- We chooose $a = -1, b = 1, c = 0$ so that goals used for previous iteration steps $\sim p_{\text{data}}$ that are in $GOID_i$ (with $y_g = 1$) have positive score $D(g) \mapsto 1$, while those too hard or too easy $y_g = 0$ have negative scores $D(g) \mapsto -1$.
- The last term in the discriminator loss is the usual discrimination term for data generated by the generator G. The G loss let as usual the generator train to fool D.

The main algorithm

- Initialize the policy, and empty set of old goals $goals_{\text{old}}$ and a Generative Adversarial Network (more on this later).
- for N total goal learning iterations:
  1. sample noise z from some distribution p_z.
  2. Set the new goals, $\frac{2}{3}$ generated by the generator of the GAN using the z noise and $\frac{1}{3}$ taken from $goals_{\text{old}}$.
  3. Update the policy to π_i using the goals above.
  4. Label the goals used.
  5. Train the GAN using the above goals and their labels
  6. Merge the old goals with the new reachable ones.

Bibliografia I

 Carlos Florensa, David Held, Xinyang Geng, and Pieter Abbeel. Automatic goal generation for reinforcement learning agents, 2017.

Grazie per l'attenzione