Automatic Goal Generation for Reinforcement Learning Agents

Alessandro Trenta - mat. 566072

ISPR - Dep. of Informatics
Università di Pisa

30 maggio 2022

The problem

- In this RL problem insead of maximizing the return over a single reward function we consider a range of reward functions r^g indexed with a goal $g \in \mathcal{G}$ (curriculum learning).
- We will consider each goal g to be a set of states $\mathcal{S}^g \subset \mathcal{S}$. The reward is 1 when the goal is reached:

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

- Assume $S^g = \{s \in \mathcal{S} : d(f(s), g) \leq \epsilon\}$ where f projects the states in the goal space $\mathcal{G}$ and d is a distance in $\mathcal{G}$.
- This is based on the assumptions that an agent who learned to reach some goals can learn to interpolate in between them, the policy learned is a good initializer for goals nearby and that if g is reachable there exist a policy that does it consistently.

- Given g we consider a Markov Decision Process that terminates whenever $s_t \in \mathcal{S}^g$. Let the return $R^g = \sum_{t=0}^{T} r_t^g$. This is a sparse binary random variable with value 1 if the agent gets close enough to the goal in at most 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. Our objective is to find the policy that maximizes the expected mean return over goals

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

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

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 goals that the agent can barely reach or that are too easy (other than avoiding catastrophic forgetting).
- Introduce the set of **Goals of intermediate difficulty** (for the i-th iteration):

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

  that is the goals which probability to be reached in at most T time steps with current policy π_i is in the range $[R_{\min}, R_{\max}]$.
- We want a way to generate new goals in this set efficiently. 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 creates goals g from noise vectors $z \sim p_z(\cdot)$. The discriminator has to distinguish which goals are in $GOID_i$.
- The two losses are defined as:

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

- Put $a = -1, b = 1, c = 0$ 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 have negative scores ($y_g = 0 \implies D(g) \mapsto -1$).
- The last term in the discriminator loss is the usual discrimination term for data generated by the generator G. The generator is trained to fool D with its usual loss.

The procedure

The algorithmic procedure is the following: starting from an initial policy π_0 and an empty set of $goals_{\text{old}}$ we iterate N times this steps

- Start by sampling the noise vectors z from $p_z(\cdot)$. Generate the set of current goals $goals = G(z) \cup sample(goals_{\text{old}})$: $\frac{2}{3}$ are generated from $G(z)$ while $\frac{1}{3}$ are sampled from the old goals we already trained on to avoid forgetting.
- Update the policy to π_i. This can be done with any RL policy based method (in this case a variant of TRPO).
- Calculate the empirical returns for each current goal w.r.t. the new policy π_i and obtain goal labels $y_g \in \{0, 1\}$ based on its difficulty (indicated by its belonging to $GOID_i$).
- Train the GAN with the current goals and labels (the p_{data} term above) to generate goals with appropiate difficulty. Update the set $goals_{\text{old}}$ with goals used this iteration which are at least ϵ distant from those already in $goals_{\text{old}}$ (avoid concentration).

Experimental Results

- The model was tested in 4 settings. An ant locomotor in a free bidimensional space and a U shaped maze, a single point mass in a multi-path maze and in a N dimensional space with decreasing reachable volume as $N \to \infty$.
- The goals were points in the space that the agent has to reach within ϵ radius of precision in T steps.
- Sampling goals uniformly from $\mathcal{G}$ suffers from training on currently not reachable goals. Training with $GOID_i$ goals (as in our model) gives far better results.
- The goal GAN samples new goals efficiently and intuitively. In the free setting they cover circles of increasing radius while in the maze settings they spread following the paths.
- The percentage of newly generated goals in $GOAL_i$ is every iteration around 20%: the agent always has enough goals. In the end the agent can reach with high success rate any goal.

- To label a goal it is evaluated around 4 times. This allows to choose $R_{\text{min}} \in (0, 0.25)$, $R_{\text{max}} \in (0.75, 1)$ without significant performance changes.
- To show the efficiency of the goal GAN the model was also tested against two goal selection models: one where the GAN is trained on every goal attempted in last iteration (even those with $y_g = 0$), the second samples uniformly from $\mathcal{G}$ keeping only the ones in $GOID_i$ (oracle).
- The first method performed worse as new goals are not based on current performance. The second one generates perfect goals but is very inefficient. It gives un upper bound on performance and the main model is really close to it.
- In the N dimensional setting where the reachable area is $[-5, 5] \times [-1, 1] \times [-0.3, 0.3]^{N-2}$ (the goal has to be reached within $\epsilon_N = 0.3\frac{\sqrt{N}}{\sqrt{2}}$) the model performs very close to the oracle one, while the others suffer a lot from the decreasing feasible area.

Final comments

- This model provides an easy extension to multi-reward models. It works naturally and efficiently with good resutls and it's formulated in a general way that doesn't need too much specifications.
- In all the experiments, an agent has to move in space and reach goals that are points in the space itself which is one of the simplest settings for the problem. In this case the definition of $\mathcal{G}, \mathcal{S}^g, f(s_t), d(\cdot, \cdot)$ are trivial and the model works almost perfectly.
- If goals are not "physical" points in space to reach but instead more complicated and abstract objectives which require peculiar definitions of f and d I'm not convinced enouth that this model can generalize easily.
- It has still to be seen whether this model can generalize easily on settings where goals are of different kind: (e.g. pick an object and move it around or do actions with it).

Bibliografia I

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