ISPR Midterm 4 - Automatic Goal Generation in Reinforcement Learning

Diego Arcelli - 647979

University of Pisa

Accademic Year 2021-2022

UNIVERSITÀ DI PISA

Introduction

The paper addresses the problem of automatic goal generation in RL, where instead of having just one goal, we're interested in maximizing the expected return of many reward functions, each of which is associated with a different goal $g \in \mathcal{G}$. In the paper they use as reward function $r^g(s_t, a_t, s_{t+1}) = \mathbb{1}\{s_{t+1} \in S^g\}$, which is 1 if the agent is in a state in S^g (which is the set of states for which being there satisfies goal g), otherwise is 0. In this way the expected return $R^g(\pi)$ can be seen as the probability of reaching a state in S^g in T steps (since $\mathbb{E}[\mathbb{1}\{event\}] = \mathbb{P}(event)$). If we have a distribution of goals $p_g(g)$ the objective is to find the policy $\pi^*(a_t|s_t, g) = \arg\max_\pi \{\mathbb{E}_{g \sim p_g(\cdot)} R^g(\pi)\}$ which is the policy that maximizes the average probability of success over all the goals.

Goal labeling

The distribution from which sample goals at each iteration i, is the follwing: $GOID_i = \{g \in \mathcal{G} : R_{min} \leq R^g(\pi_i) \leq R_{max}\}$, where R_{min} and R_{max} are respectively the minimum and maximum expected return for a goal g at iteration i. They use this distribution to avoid to sample infeasible goals or goals that have already been mastered. To approximate the sampling from $GOID_i$, first we estimate $y_g \in \{0, 1\}$ for all $g \in \mathcal{G}$, by computing the fraction of success among all the trajectories that reaches goal g in the previous iteration, and discard it if it's not between R_{min} and R_{max}, so $y_g = 0$ means $g \notin GOID_i$, while $y_g = 1$ means $g \in GOID_i$. Then we use those labels to train a Least Square GAN (LSGAN) to uniformly sample from $GOID_i$. A LSGAN is like a normal GAN but it uses the least squares loss to train the generator and the discriminator.

Equation

The optimization problem that we need to solve in order to train the LSGAN is the following:

$$\min_D \left\{ \mathbb{E}_{g \sim p_{data}(g)} \left[y_g (D(g) - b)^2 + (1 - y_g)(D(g) - a)^2 \right] + \mathbb{E}_{z \sim p_z(z)} \left[D(G(z) - a)^2 \right] \right\} \quad (1)$$

$$\min_G \left\{ \mathbb{E}_{z \sim p_z(z)} \left[D(G(z) - c)^2 \right] \right\} \quad (2)$$

$D(g)$ is the discriminator network which is trained to predict whether a goal is in $GOID_i$ or not. $G(z)$ is the goal generator network that is trained to produce goals from a random input vector z, so that D will classify them as goals that belong to $GOID_i$. b is the label that we want D to output if the input goal is in $GOID_i$, otherwise we want D to output a. c is the value that G wants D to produce in order to minimize G's loss (they used $b = 1$, $a = -1$ and $c = 0$). Differently from the original LSGAN formulation, they introduced the binary variables y_g (which are computed as explained in the previous slide) which allow D to learn also when g does not belong to $GOID_i$, and they call this model Goal GAN.

Equation

Equation 1 tells that we want to find the parameters of D which minimize both the error of D in recognizing if the actual goals are in $GOID_i$ or not, and the error in recognizing data generated from G as not in $GOID_i$. If $g \in GOID_i$ then $y_g = 1$ and so we optimize the term $\mathbb{E}[(D(g) - b)^2]$, which is minimized if $D(g)$ correctly outputs b. Conversely if $g \notin GOID_i$ and so $y_g = 0$, we optimize the term $\mathbb{E}[(D(g) - a)^2]$, which is minimized if $D(g)$ correctly outputs a. Without introducing y_g, we would only have $\mathbb{E}[(D(g) - b)^2]$ in equation 1, and so D couldn't have learned from goals which are not in $GOID_i$.
Equation 2 tells us that we want to find the parameters of G which minimize the error of D in predicting if the goals generated by G are in $GOID_i$ or not. To minimize the loss, we train the parameters of G to push D to output c.

Policy training

After the GAN is trained the policy optimization phase can be made, using any policy optimization algorithm (in th paper the used the Trust Region Policy Optimization). So in the end we'll have an iterative algorithm that at each iteration i:

1. Samples goals from $GOID_i$ using the Goal GAN trained at iteration $i - 1$
2. Updates the policy using TRPO and computes the return for each reward
3. Uses the returns to compute the labels y_g for each goal g
4. Uses the labels to train the Goal GAN to approximate the sampling of $GOID_{i+1}$

Experimental results

The authors test their method in ant locomotive problem, where a quadruped agent has to move first in a free space, and then in a U-shaped maze, in order to arrive from a starting position to another one. The goals are the (x, y) position of the center on mass of the agent (so a 2-dimensional goal space). Therefore for a goal g the reward is 1 if the CoM position of the agent is close to the position in the space associated with that goal, otherwise is 0. Their experiments show that their Goal GAN yields to a faster maximization, than other methods in the literature. Then they analyze the 2-dimensional goal space at each iteration, to check the states generated by the Goal GAN, and they conclude that the model is able to generate goals of the appropriate difficulty, which means that the goals in $GOID_i$ are those nearby the current position of the agent.

Experimental results

Then they show that their Goal GAN can track a multi-modal distribution by training a point mass agent (which means that the agent can be seen as a point in the space) in a multi-path maze, where the agent is able to reach an high maximization of the coverage.
Finally, to test if the method can scale to higher dimensional goal spaces, they test it on a n-dimensional point mass agent which has to move in an hypercube, however just a subset of the hypercube is reachable from the agent, since in many real world RL applications the set of feasible states is a small subset of the full state space.
The experiments show that compared to other methods, their Goal GAN is able to keep an high maximization of the coverage as n increases.

Comments

I found interesting the idea of generating goals by defining the distribution of goals of intermediate difficulties, so that they could train a model which is able to generate only feasible goals for the agent in a specific moment. I also found interesting how they defined the loss functions of the GAN, with the introduction of the labels y_g, so that the discriminator is not only trained to recognize goals from $p_{data}(g)$ as real, and goals from $p_z(z)$ as fake, but is trained to recognize whether goals from $p_{data}(g)$ are in $GOID_i$ or not, and to recognize goals from $p_z(z)$ as not in $GOID_i$. With this definition the discriminator can be trained on both positive and negative samples from $p_{data}(g)$.

Bibliography

[1] Carlos Florensa, David Held, Xinyang Geng, and Pieter Abbeel.
Automatic goal generation for reinforcement learning agents.
In *International conference on machine learning*, pages 1515–1528. PMLR, 2018.

[2] Xudong Mao, Qing Li, Haoran Xie, Raymond YK Lau, Zhen Wang, and Stephen Paul Smolley.
Least squares generative adversarial networks.
In *Proceedings of the IEEE international conference on computer vision*, pages 2794–2802, 2017.

[3] Yan Duan, Xi Chen, Rein Houthooft, John Schulman, and Pieter Abbeel.
Benchmarking deep reinforcement learning for continuous control.
In *International conference on machine learning*, pages 1329–1338. PMLR, 2016.