UNIVERSITÀ DI PISA

# Automatic Goal Generation for Reinforcement Learning Agents

- C. Florensa , D. Held, X. Geng , P. Abbeel (2018)

**4° MIDTERM: A brief overview of paper n°15**

Irene Pisani - [redacted]

i.pisani1@studenti.unipi.it

31 May 2022

M.Sc. Computer Science, Artificial Intelligence

Intelligent System for Pattern Recognition

Academic year 2021/2022

# Introduction to the proposed framework

## STARTING PROBLEM:
### RL agent for single-task settings

Tipically, an agent that is trained with RL technique to optimize a single reward function is only capable of learning to perform a **single task**. Such an approach does not scale well to enviroments in which an agent needs to perform a collection of tasks rather than a single one.

## PROPOSED SOLUTION:
### RL agent for multi-task settings

Florensa et al. proposed an approach that allows an agent to automatically discover and learn to perform a **wide range of tasks**, without requiring any prior knowledge of its environment.

Their main contribution is a method for **automatic curriculum generation** that considerably improves the sample efficiency of learning to reach all multiple feasible goals in a multi-task settings.

This method also naturally handles sparse reward functions: it dynamically modifies the probability distribution from which goals are sampled to ensure that the generated goals are always at the appropriate difficulty level, until the agent learns to reach all goals within the feasible goal space.

## FRAMEWORK: brief outline

The starting point was to consider the problem of **maximizing the average success rate of the agent over all possible goals**. In order to efficiently maximize this objective, at every training stage the algorithm must focus on goals that are at the appropriate level of difficulty for the current policy.

- The proposed algorithm iterates through each of the following steps until the policy converges:
    1. **Goal labelling**: label a set of goals based on whether they are at the appropriate level of difficulty for the current policy.
    2. **Adversarial Goal Generation**: using these labeled goals, a generator network is trained (with adversarial training procedure) to output new goals at the appropriate level of difficulty.
    3. **Policy optimization**: use these new goals to efficiently train the policy, improving its objective.

Such a framework allows an agent to quickly learn an optimal policy that reaches all feasible goals and automatically creates a curriculum, in which, at each step, the generate goals are only slightly more difficult than the goals that the agent already knows how to achieve.

# Formal problem definition

## RL FRAMEWORK with goal-parameterized reward functions

- Given the **goal-space $\mathcal{G}$** and the **state-space $\mathcal{S}$**, consider a **range of reward functions $r^g$** parametrized by a goal $g \in \mathcal{G}$. Each goal $g$ corresponds to a set of state $S^g \subset \mathcal{S}$ s.t. $g$ is achivied when the agent is in any state $s_t \in S^g$.
- $S^g = \{s_t : d(f(s_t), g) \leq \epsilon\}$ where $f(\cdot)$ is a function that projects a state into goal space $\mathcal{G}$; $d(\cdot, \cdot)$ is a distance metric in goal space; $\epsilon$ is the acceptable tolerance that determines when the goal is reached.
- A reward function $r^g$ measures wheter the agent has reached the goal $g$ using the **indicator function** $\mathbb{I}$:

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

- The aim is to learn a **policy $\pi(a_t|s_t, g)$** s.t. for each $g \in \mathcal{G}$, $\pi$ acts optimally with respect to $r^g$ (i.e. that achieves a high reward for each goal $g$).
- By defining an MDP each episode terminates when $s_t \in S^g$. The return $R^g = \sum_{t=0}^{T} r_t^g$ is a binary random variable whose value indicates whether the agent has reached the set $S^g$ in at most $T$ time-steps.
- As the policies are goal-conditioned, the **expected return $R^g$** obtained by taking actions sampled from the policy can be expressed as the probability of success on a goal within $T$ time-steps.

(Eq. 2) $$R^g(\pi) = \mathbb{E}(\cdot \mid s_t, g)\mathbb{I}\{\exists\, t \in [1, \dots T] : s_t \in S^g\} = \mathbb{P}(\exists\, t \in [1, \dots T] : s_t \in S^g \mid \pi, g)$$

## ASSUMPTIONS

*«A policy trained on a sufficient number of goals in some area of the goal-space will learn to interpolate to other goals within that area. »*

*«A policy trained on some set of goals will provide a good initialization for learning to reach close-by goals. »*

*«If a goal is reachable, there exists a policy that does so reliably. »*

## OVERALL OBJECTIVE

- Assuming a uniform test distribution of goals $p_g(g)$ to perform well on, the overall objective is to find an **optimal policy $\pi^*$** s.t.

(Eq. 3) $$\pi^*(a_t|s_t, g) = arg\max_{\pi} \mathbb{E}_{g\sim p_g(\cdot)} R^g(\pi)$$

$\mathbb{E}_{g\sim p_g(\cdot)}$ = expectation over distribution of feasibile goals
$R^g(\pi)$ = return, probability of reaching $S^g$ under $\pi$

- This objective, also called the ***coverage***, measures the average probability of success over all goals sampled from $p_g(g)$; our aim is to maximize the coverage. However training on the goal distribution $p_g(\cdot)$ could be inefficient as many goal can be unfeasible, too hard, or already mastered for the current policy. To avoid this, training on goals of intermediate difficulty has been proposed.

# Method: Generative Goal Learning (1)

## FULL ALGHORITHM FOR POLICY OPTIMIZATION

**Algorithm 1** Generative Goal Learning

**Input:** Policy $\pi_0$
**Output:** Policy $\pi_N$
$(G, D) \leftarrow$ `initialize_GAN()`
$goals_{\text{old}} \leftarrow \varnothing$
**for** $i \leftarrow 1$ **to** $N$ **do**
  $z \leftarrow$ `sample_noise`$(p_z(\cdot))$
  $goals \leftarrow G(z) \cup$ `sample`$(goals_{\text{old}})$
  $\pi_i \leftarrow$ `update_policy`$(goals, \pi_{i-1})$
  $returns \leftarrow$ `evaluate_policy`$(goals, \pi_i)$
  $labels \leftarrow$ `label_goals`$(returns)$
  $(G, D) \leftarrow$ `train_GAN`$(goals, labels, G, D)$
  $goals_{\text{old}} \leftarrow$ `update_replay`$(goals)$
**end for**

At each iteration $i$:

1. Firstly, it sample a noise vector $z$ from $P_z(\cdot)$ .
2. It generate a set of goals $g$ consisting of 2/3 goals generated by **"goal generator" neural network** $G(z)$ from the noise vector $z$ and 1/3 goals sampled from goals generated during previous iterations and already saved into a replay buffer.
3. This set of goals is used to train and update the policy using RL, with the reward function given by Eq. 1. TRPO with GAE (Schulman et al., 2015) is the used RL training algorithm. The old goals are also used to train the policy, so that the agent does not forget how to achieve goals that it has previously learned.
4. Evaluate the policy by measuring policy's empirical performance on these goals - i.e. computing the return $R^g(\pi_i)$-.
5. $R^g(\pi_i)$ is used to determine each goal's label $y_g$. Estimate the label $y_g \in \{0, 1\}$ of a goal $g$ means indicating whether g $\in$ $GOID_i$ by computing:

$$y_g = \mathbb{I}\{R_{min} \leq R^g(\pi_i) \leq R_{max}\}$$

[...] Then $y_g$ is used to train a generative model from where it's possible to sample goals to train on the next iteration.

### What is *GOID* ? Why is it useful?

In order to train the policy on goals that allows to efficiently maximize the *coverage*, it is convennient to approximate the sampling of news goals from the set of **Goals of Intermediate Difficulty (*GOID*)**.

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

The described algorithm forces the generator network to ouput goals in $GOID_i$; this restriction allows to train the policy on:

- Goals $g$ for which $\pi_i$ is able to receive some min expected return. By asking for $R_{min} \leq R^g(\pi_i)$ the agent can receive enough reward signal for learning.
- Goals $g$ that still need improvement. By asking for $R^g(\pi_i) \leq R_{max}$, the agent sets a max level of performance above which it prefers to concentrate on new goals.

# Method: Generative Goal Learning (2)

**Algorithm 1** Generative Goal Learning

**Input:** Policy $\pi_0$
**Output:** Policy $\pi_N$
$(G, D) \leftarrow$ `initialize_GAN()`
$goals_{\text{old}} \leftarrow \varnothing$
**for** $i \leftarrow 1$ **to** $N$ **do**
  $z \leftarrow$ `sample_noise`$(p_z(\cdot))$
  $goals \leftarrow G(z) \cup$ `sample`$(goals_{\text{old}})$
  $\pi_i \leftarrow$ `update_policy`$(goals, \pi_{i-1})$
  $returns \leftarrow$ `evaluate_policy`$(goals, \pi_i)$
  $labels \leftarrow$ `label_goals`$(returns)$
  $(G, D) \leftarrow$ `train_GAN`$(goals, labels, G, D)$
  $goals_{\text{old}} \leftarrow$ `update_replay`$(goals)$
**end for**

## Goal GAN adversarial training procedure

This procedure is introduced to enable generative model training both with positive examples ( $y_g = 1$) sampled from the distribution to be approximated and negative examples ($y_g = 0$) sampled from a distribution that does not match the desired one.

At each iteration $i$: [...]

6. $G(z)$ is updated to uniformly output goals in $GOID_i$, for which the current policy $\pi_i$ obtains an intermediate level of return. $G(z)$ is trained using a second **"goal discriminator" network** $D(g)$ that is trained to distinguish goals $g \in GOID_i$ from $g \notin GOID_i$. $G(z)$ and $D(g)$ are optimized with modified Least-Squares GAN method (Mao et al., 2017) :

$$\min_D V(D) = \mathbb{E}_{g \sim P_{data}(g)}\left[y_g (D(g) - b)^2 + (1 - y_g)(D(g) - a)^2\right] + \mathbb{E}_{z \sim P_z(z)}\left[\left(D\left(G(z)\right) - a\right)^2\right] \quad \text{(Eq. 5)}$$

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

↳ If in second term $y_g = 1$ all value function is equal to the one in the original LSGAN framework (where there are only the first and third term).

LSGAN approach is able to generate samples that are closer to real data distribution as the least squares loss function is able to move the fake samples toward the decision boundary, penalizing samples that lie in a long way on the correct side of the decision boundary. The proposed LSGAN hyper-parameter scheme is ($a$ =-1, $b$ =1, $c$ =0), where $a$ is the target for real data, $b$ for fake data and $c$ denotes the value that $G$ wants $D$ to believe for fake data. Goal GAN uses the same hyper-parameter scheme.

However, the difference between LSGAN and Goal GAN is that $D$ is trained with both positive and negative example from $P_{data}(g)$ that could have label $y_g = \{0, 1\}$. This is done by minimizing the sum of the expected distance between the discriminator's output over goal $g \sim P_{data}(g)$ with the corresponding target ($a$ or $b$) and the expected distance between the discriminator's ouput over goal generated from $G(z)$ with target for real data ($a$ ). When 2° term disappear D is trained to discriminate between goals from $P_{data}(g)$ with $y_g = 1$ and the generated goals $G(z)$; instead, when 1° term disappear the discriminator is trained with negative examples ( $y_g = 0$ ) which the generator should not generate.

The generator is trained to fool the discriminator by output goals that match the distribution of goals in $P_{data}(g)$ for which $y_g = 1$. This is done by minimizing the expectation over $z \sim P_z(z)$ of the distance between the discriminator's output over the generated goal $G(z)$ and $c$ value.

The generated goals from the previous iteration are used to compute the Monte Carlo estimate of the expectations w.r.t. the distribution $P_{data}(g)$ .

7. Goals genereted in the current iteration are stored in the replay buffer and some of them will be used in next iteration, along with new ones, to update the policy.

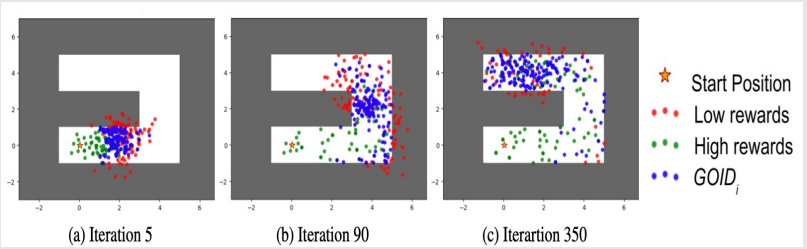
By training on goals within $GOID_i$ produced by the goal generator, this method efficiently finds a policy that optimizes the coverage objective.

**This algorithm naturally creates a curriculum as a by-product via policy optimization, without requiring any prior knowledge of the environment or the tasks that the agent must perform.**

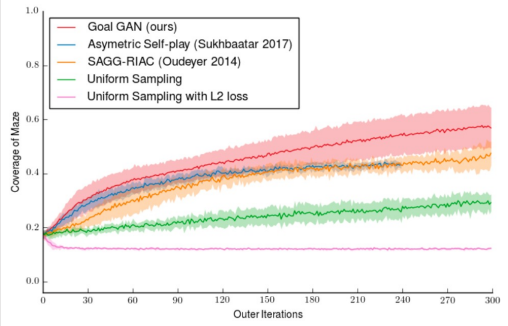
# Experimental results (1)

This method has been demostrated on 2 robotic locomotion tasks, where the goals are the (x, y) position of the Center of Mass of a dynamically complex quadruped agent.
Results reported here are referred to task 2, but same results has been obtained on task 1.

**TASK 1. Free Ant Locomotion** ⇒ Reach any given point within a free square.
**TASK 2. Maze Ant Locomotion** ⇒ Reach any given point within a U-shaped maze.

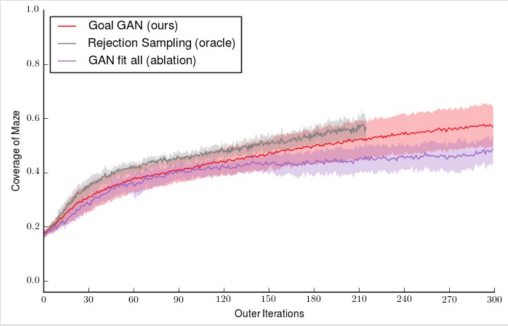



Goal GAN method's advantages were highlighted by **comparing its learning performance** with other ones.




- **Uniform Sampling** baseline has poor performance due to attempting to train on infeasible goals. By adding **L2 loss** the agent falls into a poor local optima.
- **Asymmetric Self-play** method needs a not showed additional burden to train the goal-generating policy at every outer iteration. **SAGG-RIAC** maintains an ever-growing partition of the goal-space that lead to reduced exploration and slower expansion of the policy's capabilities .

**This method yield faster maximization of the coverage objective.**




- The variant **GAN fit all**, that disregards the labels, performs worse because the expansion of the goals is not related to the current performance of the policy.
- **Rejection Sampling** train a policy on goals sampled uniformly from $GOID_i$ . This variant is orders of magnitude more sample inefficient and gives an upper bound on the performance: Goal GAN's performance are quite close to this much inefficient one.

**Efficient learning depends on genereting goals in $GOID_i$.**

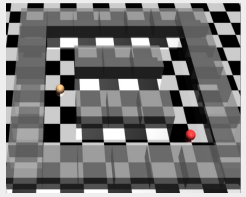
Studing the shift in the probability distribution generated by the Goal GAN along with the improvement of the policy coverage, it's possible to claim that the location of the generated goals shifts to different parts of the maze, concentrating on the area where the current policy is receiving some learning signal but needs more improvement.

The percentage of generated goals that are at the appropriate level of difficulty (in $GOID_i$ ) stays around 20% even as the policy improves.

**Goal GAN dynamically shift to sample goals of the appropriate difficulty.**

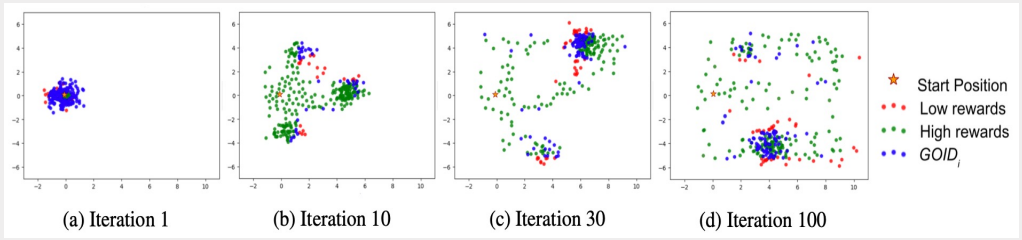
# Experimental results (2)

## TASK 3. Point-mass agent in multi-path maze

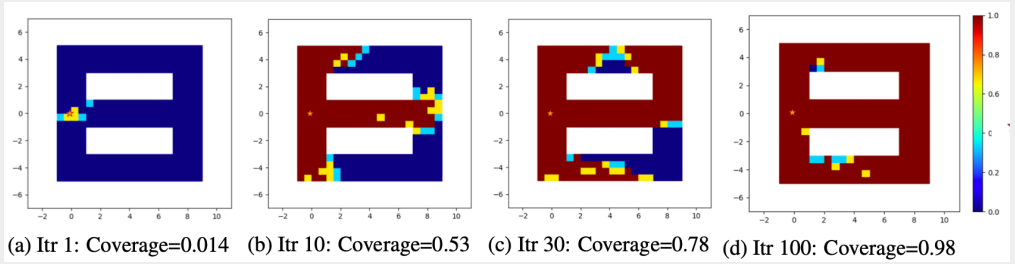


A point-mass agent is trained to reach any feasible goal corresponding to $\epsilon$-balls in state-space within a multi-path maze .

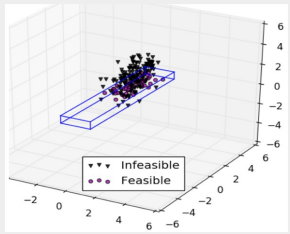
Showing the generated goal used to train the agent it can be observed that this **method produces a multi-modal distribution over goals, tracking all the areas where goals are at the appropriate level of difficulty** (i.e. $g \in GOID_i$ ).




Having several possible paths to reach a specific goal does not affect the learning of the algorithm that consistently reaches full coverage.



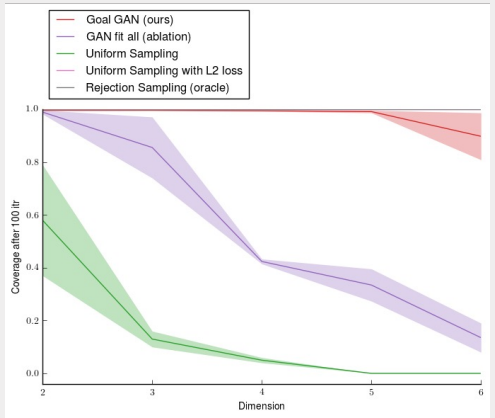

## TASK 4. N-Dimensional point-mass




N-dimension point-mass is used to show how the method scales with the dimension of the goal-space in an environment where the feasible region is kept of approximately constant volume in an embedding space that grows in dimension.

The full state-space is an hypercube, but the point-mass can only move within a small subset of this state-space; the fraction of the volume of the feasible space decreases as N increases.

The figure beside describes how the coverage value after 100 iterations changes as the dimension increases for different methods.




- **Uniform Sampling** has poor performance as the fraction of feasible states within the full state space decreases as the dimension increases. (i.e. poor learning signal).
- **GAN fit all** suffers from the increase in dimension because it is not encouraged to track the narrow feasible region.
- **Rejection Sampling Oracle** and **Uniform Sampling with an L2** have perfect performance since the optimal policy is just to go in a straight line towards the goal.

**Goal GAN's performance does not decay as much as the state space dimension increases, because it always generates goals within the feasible portion of the state space.**

# Conclusions

This method for automatic curriculum generation that dynamically adapts to the current performance of the agent solve the RL paradigm where the objective is to train a single policy to succeed on a variety of goals.

## STRONG POINT + additional considerantions

- The choice of **GANs** for goal generation is important for enable training from negative examples as well as for scaling up this approach to goal generation in high-dimensional goal spaces.
- The **LSGAN** approach gives considerable improvement in training stability.
- The introduced **Goal GAN** adversarial training procedure improves the accuracy of the generative model despite being trained with few positive samples.
- As already mentioned, the curriculum is obtained without any prior knowledge of the environment or of the tasks being performed.

## WEAK POINT w.r.t. assumptions

- A **continuous goal-space representation is required** s.t. a goal-conditioned policy can efficiently generalize over the goals.

## NOVELTIES w.r.t. related work

- This approach trains policies that learn to achieve multiple goals directly: it does not require a **skill learning** pre-training step from which useful primitives are obtained, later properly composed and used to achieve other tasks.
- Other frameworks that uses automatic **curriculum learning** can only handle a finite set of tasks and cannot handle sparse rewards. This method trains a policy that generalizes to a set of continuously parameterized tasks, and it performs well even under sparse rewards by not allocating training effort to too hard tasks.
- Frameworks involving **learning with an intrinsically specified objective** don't have an explicit notion of which states are hard for the learner. In contrast, this formulation motivates the agent to train on tasks that push the boundaries of its capabilities.

## FUTURE DEVELOPMENTS

- Select in better ways the next goal to train on by combining this goal-proposing strategy with recent multi-goal approaches.
- Focus on building hierarchy on top of the multi-task policy obtained with this method by training a higher-level policy that outputs the goal for the lower level multi-task policy.

## References

**Florensa et al., *Automatic Goal Generation for Reinforcement Learning Agents* , 2018.**
Mao et al., *On the effectiveness of least squares generative adversarial networks*, 2017.
Schulman et al., *High-Dimensional continuous control using generalized advantage estimation*, 2015.