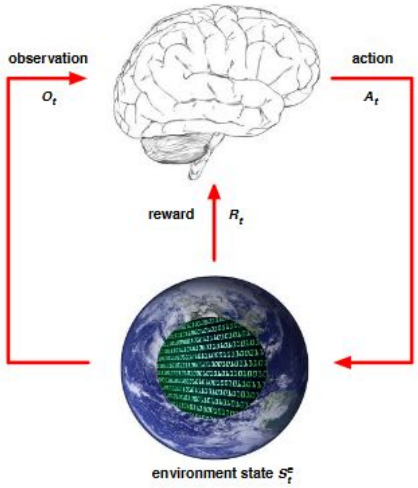
Automatic Goal Generation for Reinforcement Learning Agents

Nikita Parate
Ankit Wahane
Ponsuganth Ilangovan

What is Reinforcement Learning

- Reinforcement learning is an area of machine learning concerned with how software agents ought to take actions in an environment so as to maximize some notion of cumulative reward



Policy and Value function based learning

- A value function tells us what is the expected sum of rewards given a state and an action (i.e) expectation of the cumulative sum of rewards given a state and an action.
- Policy function maps a state to an action. It assigns a probability distribution over all actions given a state.
- Given policy $\pi_\theta(s,a)$ with parameters θ, find best θ

Objective

- A RL agent is trained to perform a single task using a single reward function but it doesn't scale up
- Sparse reward problem : When a RL agent learns from a sparse rewards, it either wins or loses and has no intermediate rewards
- A method is proposed to efficiently train a policy to achieve all possible goals
  - Discover all feasible goals
  - Focus on goals currently giving better learning

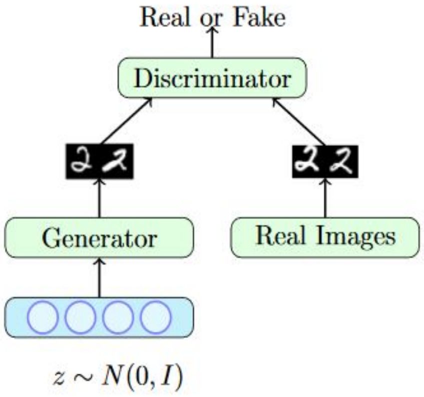
Goal Parametrized Reward Function

- The agent in state $s_t \in S \subseteq R_n$ takes an action $a_t \in A \subseteq R_m$, according to some policy $\pi(a_t|s_t)$
- Taking this action causes the agent to enter into a new state s_{t+1} according to a transition distribution $p(s_{t+1}|s_t,a_t)$, and receive a reward $r_t=r(s_t,a_t,s_{t+1})$.
- Reward function that measures whether the agent has reached the goal:
    - $r^g(s_t,a_t,s_{t+1}) = \mathbf{1}\{s_{t+1} \in S_g\}$

Challenges and Approach

- Training directly on the goal distribution $g \sim p_g(.)$ is not efficient
  - Many goals might be infeasible or too hard for the current policy π_i
  - Other goals might already be mastered by the current policy π_i
  - Solving some goals first might help for others
- Instead, train on Goals Of Intermediate Difficulty for π_i
  - $GOID_i := \{g: R_{min} \leq R_g(\pi_i) \leq R_{max}\} \subseteq G$

Generative and Adversarial Networks



- In GANs, the idea is to sample from a simple distribution (say,z~N(0,I)) and then learn a complex transformation from this to the training distribution.
- We use a “goal generator” neural network G(z)to generate goals g from a noise vector z
- We train G(z) to uniformly output goals in GOIDi using a second “goal discriminator” network D(g) which distinguish goals that are in and not in $GOID_i$

LSGAN

- When updating the generator, sigmoid cross entropy loss function will cause the problem of vanishing gradients.
- To remedy this LSGAN uses least square loss function
    - ^

$$\min_D V(D) = \mathbb{E}_{g \sim p_{\text{data}}(g)} \Big[y_g (D(g) - b)^2 +$$

$$(1 - y_g)(D(g) - a)^2 \Big] + \mathbb{E}_{z \sim p_z(z)} [(D(G(z)) -$$

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

Where a is the label for fake data (-1)

b is the label for real data (1)

Algorithm

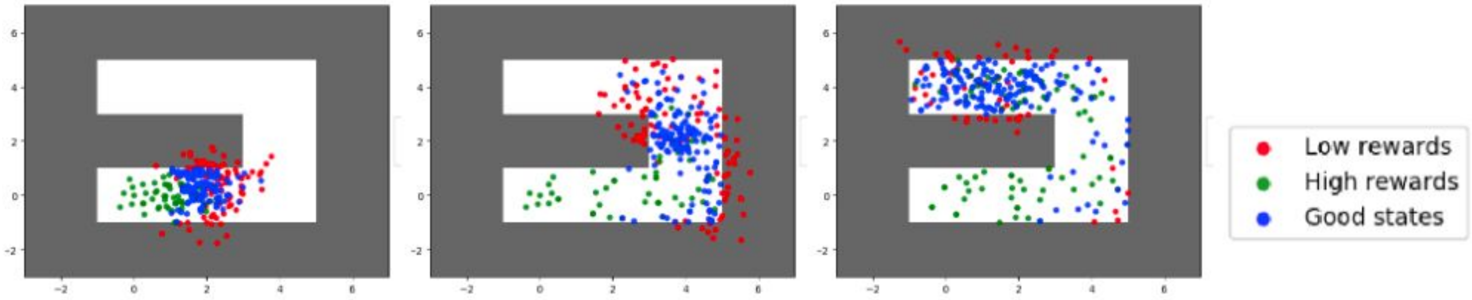
Algorithm 1 Generative Goal Learning

Input: Policy π_0
Output: Policy π_N
$(G, D) \leftarrow \texttt{initialize\_GAN}()$
$goals_{\text{old}} \leftarrow \varnothing$
for $i \leftarrow 1$ **to** N **do**
  $z \leftarrow \texttt{sample\_noise}(p_z(\cdot))$
  $goals \leftarrow G(z) \cup \texttt{sample}(goals_{\text{old}})$
  $\pi_i \leftarrow \texttt{update\_policy}(goals, \pi_{i-1})$
  $returns \leftarrow \texttt{evaluate\_policy}(goals, \pi_i)$
  $labels \leftarrow \texttt{label\_goals}(returns)$
  $(G, D) \leftarrow \texttt{train\_GAN}(goals, labels, G, D)$
  $goals_{\text{old}} \leftarrow \texttt{update\_replay}(goals)$
end for

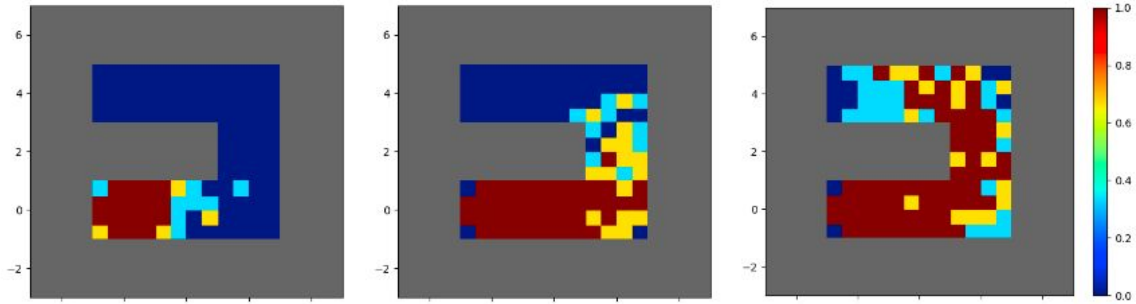
Experiment 1: Maze Ant Locomotion

- Objective is to train an ant to learn the U shape maze
- The mujoco environment for the experiment which is a physics simulation environment
- The maze with ant setting is used which gives 8 dimensional action space involving joints and positional data.
- Goals are (x,y) position of the center of mass of the ant agent.

Maze Ant Results



(a) Iteration 5 (b) Iteration 86 (c) Iterartion 350

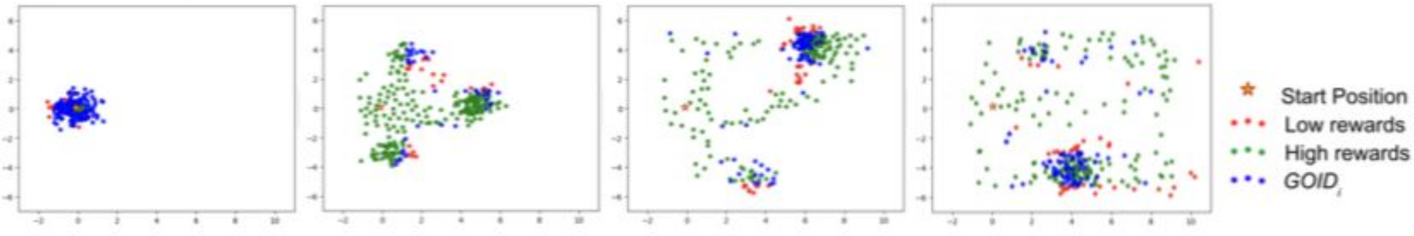


(a) Iteration 5: Coverage = 0.20 (b) Iteration 86: Coverage = 0.48 (c) Iteration 350: Coverage = 0.71

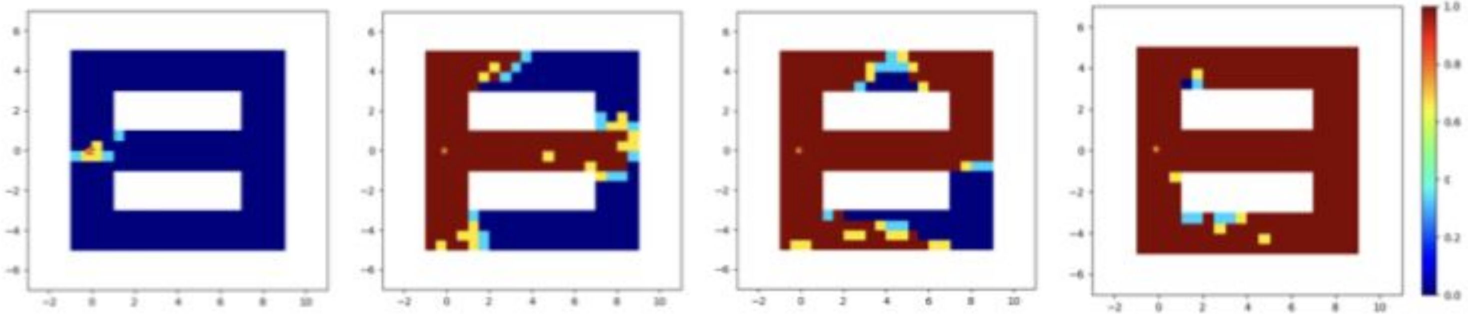
Experiment 2: Ant Multi-path Locomotion

- Objective : The ant is trained on a multi path maze
- The environment was updated according to the structure of the maze.
- Number of iterations is increases since the area to be covered by the agent has increased.

Ant Multi-path Locomotion Result

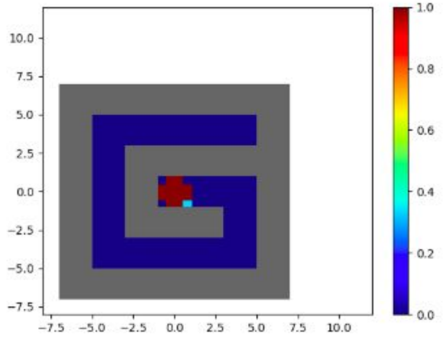


(a) Iteration 1 (b) Iteration 10 (c) Iteration 30 (d) Iteration 100

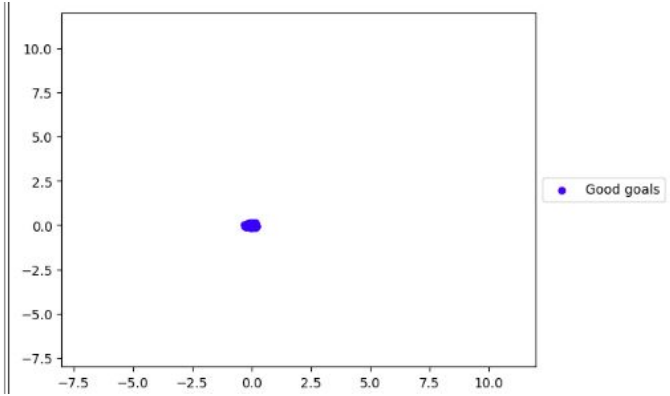


(a) Itr 1: Coverage=0.014 (b) Itr 10: Coverage=0.53 (c) Itr 30: Coverage=0.78 (d) Itr 100: Coverage=0.98

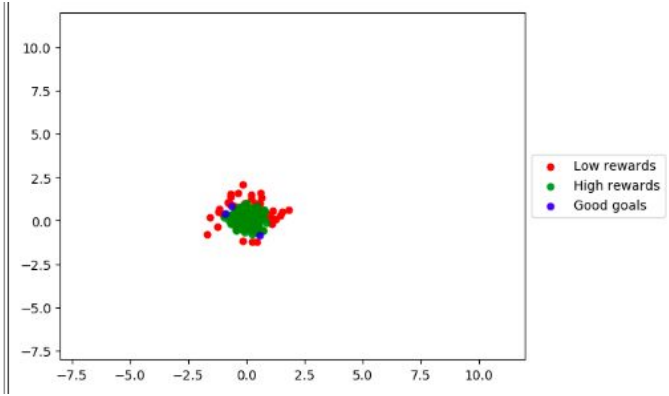
Spiral Maze



10.0
7.5
5.0
2.5
0.0
−2.5
−5.0
−7.5
−7.5 −5.0 −2.5 0.0 2.5 5.0 7.5 10.0
1.0
0.8
0.6
0.4
0.2
0.0



Good goals
10.0
7.5
5.0
2.5
0.0
−2.5
−5.0
−7.5
−7.5 −5.0 −2.5 0.0 2.5 5.0 7.5 10.0



Low rewards
High rewards
Good goals
10.0
7.5
5.0
2.5
0.0
−2.5
−5.0
−7.5
−7.5 −5.0 −2.5 0.0 2.5 5.0 7.5 10.0

Conclusion

- This proposed RL algorithm trains a single policy on a variety of goals, under sparse rewards.
- Since the curriculum is automatic, it dynamically adapts to the current performance of the agent
- We used GAN to automatically generate goals for our policy that are always at a appropriate level of difficulty

THANK YOU!