Hide an Escape agent using Imitation Learning

Yunao Shen
Rutgers University
Piscataway, New Jersey
ys670@rutgers.edu

Ruolin Qu
Rutgers University
Piscataway, New Jersey
rq40@rutgers.edu

Shengdong Liu
Rutgers University
Piscataway, New Jersey
sl1563@rutgers.edu

ABSTRACT

This project proposes modeling authentic chasing and escaping behaviors for multi-agents in an enclosed environment where the agents play a competitive game with some interactive objects. We are using deep reinforcement learning combined with imitation learning to train agents to acquire emergent strategies such as tool use, where agents can change the environment for reaching goals. We first use imitation learning to bootstrap the agent's policy using behavior from a player and train them to do some simple strategies such as chasing, hiding and using tools. After that, the agents can learn to change their environment to catch the goals.

KEYWORDS

Multi-agent, Unity ML-Tool kit,Imitation learning, Reinforcement Learning, GAIL

ACM Reference Format:
Yunao Shen, Ruolin Qu, and Shengdong Liu. 2020. Hide an Escape agent using Imitation Learning. In *Proceedings of ACM Conference (Conference'17).* ACM, New York, NY, USA, 5 pages. https://doi.org/10.1145/nnnnnnn.nnnnnnn

1 INTRODUCTION

Problem Statement. Many situations in real world require artificial agents.However, It is challenging to implement competitive and cooperative behavior in AI agent. Some modern computer games attempt to simulate such behavior, in which player and non-player hostiles compete against each other to avoid being detected or to catch others.However,their movements are pre-defined by behavior trees.With all commands being completely settled by scripts, those characters will only conduct rigid behavior with same pattern such as following the player or moving along a fixed routine.They cannot dynamically adapt to player's behavior and evolve corresponding strategies, which limits their performance in a real-world application.

Motivation and Applications. We wish to implement an AI that is capable of exerting complex strategies by racing between agents as well as learning player's behavior in a computer-simulated situation. We are trying to solve this problem with some advanced machine learning methods including deep reinforcement learning, imitation learning and GAIL with limited resources and we also want to explore further for AI in an imperfect information game.

Conference'17, July 2017, Washington, DC, USA

ACM ISBN 978-x-xxxx-xxxx-x/YY/MM.
https://doi.org/10.1145/nnnnnnn.nnnnnnn

Challenges. Adaption to dynamic environment. Strategies should adapt to complicate and changeable obstacles, such as S-shape channels that may cause dead loop. Multi-agent cooperation strategies(baiting, intercepting), Which requires multi-agent training method where different agents have relative rewards, states and actions. Reinforcement learning with combined rewards. A single reward policy is clear and straightforward to guide the escapee to the exit, but is limited in providing feedback of valuable strategies to the algorithm. Multiple behavioral reward, on the other hand, strengthen beneficial short-term strategies while overlooking the primary objective of reaching the exit.

Technical Innovations and Contributions . The combination of Imitation Learning and Deep reinforcement learning can dramatically reduce the time the agents take to solve the environment. What are the main contributions of your research? **Main Deliverable**. open-sourced environments and code for environment construction. playable games for real person compete against agents. What is the main deliverable of your research ?

2 RELATED WORK

Unity[4]. provides a new open source toolkit for creating and interacting with simulation environments using the Unity platform: Unity ML-Agents Toolkit. By taking advantage of Unity as a simulation platform, the toolkit enables the development of learning environments which are rich in sensory and physical complexity, provide compelling cognitive challenges, and support dynamic multi-agent interaction. Liu et al. introduced agents in soccer ball and discover some strategies like intercepting and ball passing. In solving hide-seek problems,a team from OpenAI [1] provides a hide and seek programs aimed on the cooperation and environment interaction. The hider-agents could create shelter with cooperation.Through only a visibility-based reward function and competition, agents learn many emergent skills and strategies including collaborative tool use. However, the training requires large order of magnitude episodes. We try to use previous imitation learning to reduce the training time. Deepmind AlphaStar[6] is the first agent that reach the highest league of human players in a widespread professional e-sport without simplification of the game,however it consists of tens of thousands of time-steps and thousands of actions, observation includes all visible units and attributes. Takes about 50 days training to reach that goal. and AlphaStar still can not even cause challenges for the top human player with some obvious defect in strategies.

3 METHODS

Training is performed with a combination of deep reinforcement learning(DRL) [2] and Generative Adversarial Imitation Learning (GAIL)[3].

1. DRL The goal of reinforcement learning is to learn a policy, which is essentially a mapping from observations to actions.
Extrinsic procedure a)Observation. Agent's observation of environment as well as its opponent is based on simulation of view by having several rays emit from the agent. If the rays hit any object(the other agents, walls, etc), values indicating the distance and position will be returned to the agent. b)Action. Agent's action space is move forward, move backward, turn left and turn right. After each action the agent will transit to another state with new observation. c).Reward. Reward is evaluated by current state.the escapee win 1 point if it arrives the exit and lose 1 point if it is caught or time up. The chaser has the converse reward.
Generative Adversarial Imitation Learning
Intrinsic procedure 2. GAIL Generative Adversarial Imitation Learning[3], is a new general framework for directly extracting a policy from data. It is used to solve the problem of sparse reward in DRL.

In this framework, a neural network, the discriminator, is taught to distinguish whether an observation/action is from a demonstration or produced by the agent. This discriminator can the examine a new observation/action and provide it a reward based on how close it believes this new observation/action is to the provided demonstrations. In the training process, for each action the agent make, the discriminator produce a reward based on how similar the action is to the actions in the demo. After getting the reward received from GAIL, we send it as one of the intrinsic reward signal to the reinforcement learning model. Then the agent would try to maximize the reward, thus encouraging the agent to mimic the behaviors in the demo. As the agent making actions more and more similar to the demo actions, the discriminator also keeps training on those more similar actions. Thus the discriminator is becoming more and more accurate.

4 TRAINING EVIRONMENT

The training environment is a randomly generated playground. It would reset in the beginning of each round.When resets,all the walls are deactivated. Then each wall(except the boundaries) in the playground has a probability of 75% to be activated.The escapee would randomly spawn at one of the four spawn locations.The chaser would randomly spawn at one of the four spawn locations.

5 TRAINING PROCESS

We build an alternately learning structure to alternately train escapee and chaser one by one in each section. For both agents, they start from easier circumstance to harder circumstance.

Training Section	Train_1	Train_2	Train_3	Train_4	Train_5
Training Parameters					
Escapee Speed	2.0	2.0	2.0	2.0	2.0
Chaser Speed	2.0	2.5	2.25	2.25	2.5
Training Target					
Training Model	Escapee	ChaserEasy	EscapeeEasy	ChaserHard	EscapeeHard
Against Model	ScriptChaser	EscapeeBasic	ChaserEasy	EscapeeEasy	ChaserHard

Figure 1: the Alternately Learning Structure

a) Training section 1: To get the first AI agent, we train the basic escapee based on the script-driven chaser. We set the agents speed the same to help boost training. We record 5 demos as escapee playing against the ScriptChaser.

Training Section	Train_1	…
Training Parameters		
Escapee Speed	2.0	…
Chaser Speed	2.0	…
Training Target		
Training Model	EscapeeBasic	…
Against Model	ScriptChaser	…

Figure 2: Training 1 settings

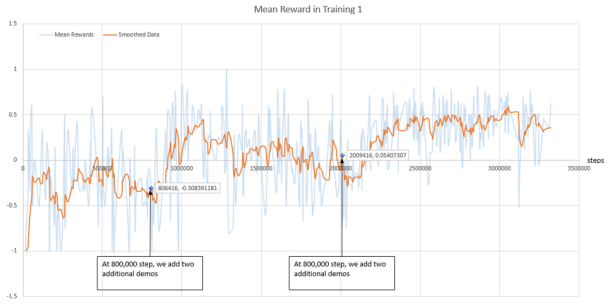


Figure 3: Training 1 mean reward chart in training

After about 3,300,000 steps we get the AI escapee agent: EscapeeBasic. We test EscapeeBasic for 1000 rounds, the win rates is 32.4%

b) Training section 2: We train the first AI chaser based on EscapeeBasic. It's the first time we train chaser, so we set it speed to 2.5 which would be easy for chaser to start learning with. We record 5 demos as chaser playing against EscapeeBasic.

Training Section	Train_1	Train_2
Training Parameters		
Escapee Speed	2.0	2.0
Chaser Speed	2.0	2.5
Training Target		
Training Model	EscapeeBasic	ChaserEasy
Against Model	ScriptChaser	EscapeeBasic

Figure 4: Training 2 settings

After about 2,200,000 steps we get the AI chaser agent: ChaserEasy. We test ChaserEasy for 1000 rounds, the win rates is 55.7%

c) Training section 3: Continue with EscapeeBasic, We train the escapee further against ChaserEasy. This is the first training section for escapee to play against AI chaser, so we set chaser speed lower to boost training. We record 4 demos as escapee playing against ChaserEasy

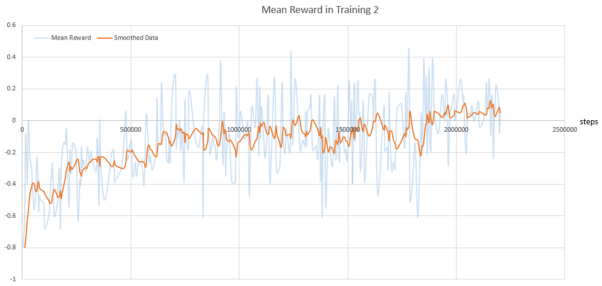


Figure 5: Training 2 mean reward chart in training

Training Section	Train_2	Train_3
Training Parameters		
Escapee Speed	2.0	2.0
Chaser Speed	2.5	2.25
Training Target		
Training Model	ChaserEasy	EscapeeEasy
Against Model	EscapeeBasic	ChaserEasy

Figure 6: Training 3 settings

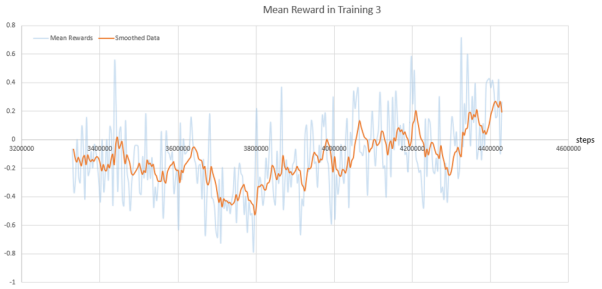


Figure 7: Training 3 mean reward chart in training

After about 1,100,000 more steps we get EscapeeEasy trained from EscapeeBasic. We test EscapeeEasy for 1000 rounds, the win rates is 65.8%

d) Training section 4: Continue with ChaserEasy, We train the chaser further against EscapeeEasy. Comparing with Train 2 (last time we train chaser), the chaser agent will face harder conditions: Lower agent speed and Stronger opponent(better trained agent). We record 5 demos as chaser playing against EscapeeEasy.

After about 4,400,000 more steps we get ChaserHard trained from ChaserEasy. We test ChaserHard for 1000 rounds, the win rates is 50.7%

e) Training section 5: Continue with EscapeeEasy, We train the escapee further against ChaserHard. Comparing with Train 3 (last time we train escapee), the escapee agent will face harder conditions: Higher opponent speed and Stronger opponent(better trained agent). We record 4 demos as escapee playing against ChaserHard.

Training Section	Train_2	Train_4
Training Parameters		
Escapee Speed	2.0	2.0
Chaser Speed	2.5	2.25
Training Target		
Training Model	ChaserEasy	ChaserHard
Against Model	EscapeeBasic	EscapeeEasy

Figure 8: Training 4 settings

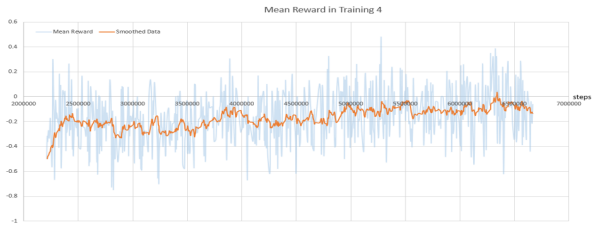


Figure 9: Training 4 mean reward chart in training

Training Section	Train_3	Train_5
Training Parameters		
Escapee Speed	2.0	2.0
Chaser Speed	2.25	2.5
Training Target		
Training Model	EscapeeEasy	EscapeeHard
Against Model	ChaserEasy	ChaserHard

Figure 10: Training 5 settings

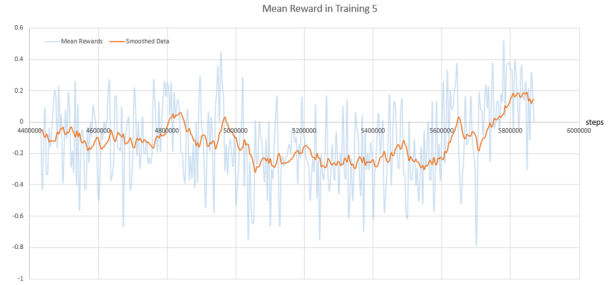


Figure 11: Training 5 mean reward chart in training

After about 1,600,000 more steps Now we get EscapeeHard trained from EscapeeEasy. We test EscapeeHard for 1000 rounds, the win rates is 46.6%

In all training sections, we collect the win rates of both agents when testing after the section is over.

Training Section	Train_1	Train_2	Train_3	Train_4	Train_5
Training Parameters					
Escapee Speed	2.0	2.0	2.0	2.0	2.0
Chaser Speed	2.0	2.5	2.25	2.25	2.5
Training Target					
Training Model	Escapee	ChaserEasy	EscapeeEasy	ChaserHard	EscapeeHard
Against Model	ScriptChaser	EscapeeBasic	ChaserEasy	EscapeeEasy	ChaserHard
Win Rates After Training					
Escapee	32.4%	44.3%	65.8%	49.3%	46.6%
Chaser	67.6%	55.7%	34.2%	50.7%	53.4%

Figure 12: Win rates of all training sections

6 EVALUATION

First, we evaluate whether the agents learning better and better in the alternately learning structure. Both ChaserHard and EscapeeHard are trained from the early AI agents ChaserEasy and EscapeeEasy, without the influence of the script-driven chaser we use in training 1. So the behaviors of ChaserHard and EscapeeHard would better prove whether the alternately learning structure works. We use win rates as the evaluation.

ChaserHard is trained from ChaserEasy in training 4. Comparing with training 3, ChaserEasy and ChaserHard are under same circumstance: same speed of agent and same opponent agent. The win rate of chaser agent raise from 34.2% to 50.7%. So the chaser agent is becoming better in the alternately learning structure. EscapeeHard is trained from EscapeeEasy in training 4. In both training 4 and 5, the escapee is facing the same opponent: ChaserHard. But the agents speed is different. So we add a test group to keep same circumstance. Comparing from Training 4 and test group, the win rate of escapee raise from 49.3% to 52.9%.

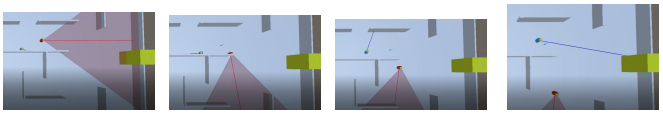
Training Section	Train_3	Train_4
Training Parameters		
Escapee Speed	2.0	2.0
Chaser Speed	2.25	2.25
Training Target		
Training Model	EscapeeEasy	ChaserHard
Against Model	ChaserEasy	EscapeeEasy
ChaserWinRate	34.2%	50.7%

Figure 13: Win rates of chaser agent in training 3 and 4

Training Section	Train_4	Train_5	Test Group
Training Parameters			
Escapee Speed	2.0	2.0	2.0
Chaser Speed	2.25	2.5	2.25
Training Target			
Training Model	ChaserHard	EscapeeHard	EscapeeHard
Against Model	EscapeeEasy	ChaserHard	ChaserHard
EscapeeWinRate	49.3%	46.6%	52.9%

Figure 14: Win rates of chaser agent in training 4 and testing

The agents exhibit some meaningful strategies after training.For example, the escapee learns to sneak behind chaser as the chaser move straightly toward one direction.Moreover, the escapee turns around and gets ready to escape before being detected when the chaser turns halfway around.15 Also, instead of following the same routine of the escapee to catch it, the chaser predict the final destination of the escapee and learns to intercept by taking the shortest path directly to that position.16



Sneak behind chaser (a) | Try to turn away to escape while chaser turning (b) | Turn Back to the exit (c) | go straight to the exit (d)

Figure 15: Escapee sneak behind Chaser

Chaser detect Escapee (a) | Chaser pick the way (b) | Chaser go to the position (c) | chaser detect escapee again (d)

Figure 16: Chaser intercept escapee

7 LIMITATION AND FUTURE WORK

1. Automatic operation. The alternately learning structure we presented in this algorithm is now implemented in a manual way, in which we switch the training of chaser and escapee by hand after observation of the win rate and agent's strategy. In the future, we plan to formalize the procedure to get rid of manual operation by adding quantitative index indicating the completeness of training.

2. Relationship diversity. For now, only hostile relationship is present in our implementation. We would like to ameliorate the algorithm to train strategies featuring cooperation between multiple agents of ally relation.

3. Additional ways of interaction. Currently the interaction is limited between agents. In the future we would like to add features in which agent behavior can affect connectivity of the map, such as doors that can be opened or walls that can be pushed. Also, we would simulate a more realistic and complicated situation with more features, such as additional moving modes or detectable walk noise.

REFERENCES

[1] Bowen Baker, Ingmar Kanitscheider, Todor Markov, Yi Wu, Glenn Powell, Bob McGrew, and Igor Mordatch. 2019. Emergent Tool Use From Multi-Agent Autocurricula. arXiv:cs.LG/1909.07528

[2] Vincent François-Lavet, Peter Henderson, Riashat Islam, Marc G. Bellemare, and Joelle Pineau. 2018. An Introduction to Deep Reinforcement Learning. *Foundations and Trends® in Machine Learning* 11, 3-4 (2018), 219–354. https://doi.org/10.1561/2200000071

[3] Jonathan Ho and Stefano Ermon. 2016. Generative Adversarial Imitation Learning. arXiv:cs.LG/1606.03476

[4] Arthur Juliani, Vincent-Pierre Berges, Esh Vckay, Yuan Gao, Hunter Henry, Marwan Mattar, and Danny Lange. 2018. Unity: A General Platform for Intelligent Agents. arXiv:cs.LG/1809.02627

[5] Siqi Liu, Guy Lever, Josh Merel, Saran Tunyasuvunakool, Nicolas Heess, and Thore Graepel. 2019. Emergent Coordination Through Competition. arXiv:cs.AI/1902.07151

[6] Oriol Vinyals, Igor Babuschkin, and Wojciech M. Czarnecki. 2019. Grandmaster level in StarCraft II using multi-agent reinforcement learning.