Bachelor Thesis - "Hide and Seek Games"

Petros Hipp

December 2022

Contents

1 Introduction

2 Pursuit-Evasion Problems

3 Baselines

In the following, we will have a look at the work of Qi et al. When regarding traditional pursuit-evasion games, oftentimes one would simplify certain properties of the environment. Examples of such simplifications are using discrete models without including kinematic constraints or omitting the possibility for obstacles in the environment. The authors now focus on the pursuit-evasion game including said constraints.
Firstly, the usual grid environment in which the agents can only move in the four cardinal directions (up, down, left and right) is replaced by a continuous environment. This way, when regarding the agent as a robot, its kinematic

properties such as velocity or turning speed influence the movement in the environment.
In addition to this, static obstacles in the environment are introduced which the agents need to maneuver around, otherwise resulting in negative rewards.
To train the pursuer and evader agents, they use curriculum deep reinforcement learning. This way, the agents start off by learning very simple tasks before moving on to increasingly complex ones. Furthermore, Qi et al. apply the self-play mechanism in the training process, meaning that the agents simultaneously learn by playing against past versions of themselves in order to improve.
Comparative results have shown that this leads to better results.

3.1 Game environment

The pursuit-evasion problem is a type of differential game in which a pursuer agent tries to capture the evader agent while the latter attempts not to be caught. In our case, the game environment is a two-dimensional space wherein the actors can move. They can do so by moving forward or backward (linear velocity) and changing directions (angular velocity). Once the distance between pursuer and evader is less than a certain threshold, the game ends. This is the same case with an agent stepping outside of the boundary of the environment, or colliding with an obstacle.

3.1.1 Rewards

Depending on the action taken, an agent receives a certain reward. There are two different reward functions for the pursuer and the evader respectively.

$$r_{pursuer} = \begin{cases} 10 & \text{if } d_t < D_{catch} \\ -10 & \text{if collides with an obstacle} \\ -10 & \text{if leaves environment} \\ c(d_{t-1} - d_t) & \text{else} \end{cases}$$

The distance between both agents at timestep t is noted as d_t and D_{catch} is a set distance between them, for which the evader is considered as being caught by the pursuer. The else-case makes the pursuer move toward the evader.

The reward function of the evader is quite similar. However, notice how the evader receives a negative reward for being caught. Furthermore, it is encouraged to be far away from the pursuer.

$$r_{pursuer} = \begin{cases} -10 & \text{if } d_t < D_{catch} \\ -10 & \text{if collides with an obstacle} \\ -10 & \text{if leaves environment} \\ c(d_t - d_{t-1}) & \text{else} \end{cases}$$

3.2 Framework

The algorithm used in this case is Deep Q-network (DQN) which is used in conjunction with Experience Replay to store transitions in an experience pool. When an agent interacts with the environment, it receives a transition out of the experience pool, where the data is randomly sampled from. In DQN, one neural network is set to estimate the current action-value Q, aswell as a similiar network with different parameters to estimate the target action-value $\hat{Q}$.

Curriculum learning is used in combination with DQN, so the agents start learning simple problems before moving on to complex ones. The training process is split into two sections. First, the pursuer is trained to move towards a set point in the environment which acts as an immobile evader. This reduces the difficulty since the noise of the samples is smaller. Second, a more intelligent pursuer is used together with an evader to train in competitive self-play. This increases robustness since both agents play with an opponent of similar intelligence and can reinforce each other.

3.3 Results of the paper

Qi et al. perform simulations to verify the effectiveness of the proposed curriculum learning. They name their pursuer/evader agents which were obtained by curriculum learning $P_{curriculum}$ and $E_{curriculum}$ respectively and their pursuer/evader agents which were obtained by self-play from scratch $P_{scratch}$ and $E_{scratch}$ respectively.

The authors show that after 500 rounds, $P_{curriculum}$ had a 70% win rate against $E_{scratch}$. In addition, $E_{curriculum}$ had a win rate of more than 70% against $P_{scratch}$.

Furthermore, an evader agent taking random actions (E_{random}) and a similar pursuer agent (P_{random}) were trained for 500.000 episodes. Qi et al. show that the training efficiency when facing agents trained by self-play is higher than when facing random opponents.

In the next experiment, the authors show that when playing against E_{random}, P_{random} and $P_{curriculum}$ have similar win rates of 91% and 93% respectively. However, compared to P_{random}, $P_{curriculum}$ mostly wins by just waiting for E_{random} to hit an obstacle instead of catching it.

After replacing E_{random} with the more intelligent evader $E_{scratch}$, results show that the win rates of P_{random} and $P_{curriculum}$ change to 43.4% and 70.6% respectively. When comparing both pursuer agents, it can be seen that $P_{curriculum}$ catches the evader more often, as well as forcing it to hit obstacles.

Based on all results of their experiments, Qi et al. infer that their training method yields superior results against opponents of certain levels of intelligence.

3.4 Extensions & Improvements

The authors state that for future work, one could focus on extending the action space into a continuous one to make the motion of the actors smoother.

Further potential extensions that could be added include the following ideas:

- One could use moving obstacles instead of static ones, as obstacles in the real world naturally might be moving too. Ideally, the agents should still be able to maneuver around the obstacles.
- In traditional pursuit-evasion scenarios, the number of pursuers is usually equal or greater than the number of evaders. It might be interesting to see the outcome when there are more evaders than pursuers. Potentially, the catch rates might then increase and the number of times when the pursuer wins by forcing the evader to hit an obstacle might decrease, since it becomes easier to catch an evader. Therefore, one might observe a change in pursuer strategies.
- Pursuer agents usually operate with identical kinematic properties such as velocity or turn rate. Individually changing these attributes for a set amount of episodes to have different versions of pursuers in the same environment could be interesting.
- The pursuers currently only receive negative rewards when colliding with obstacles (or moving out of bounds). Therefore, colliding with another pursuer is not being punished, but probably should be in a real-life scenario.
- One could implement a certain view range in which the agents can see each other, while obstacles could also obstruct vision and influence agent behaviour.

4 Experiments

5 Conclusion

6 Bibliography