Multi agent approach to the game of tag

João Porto
Instituto Superior Técnico
Lisbon, Portugal
joaofporto@tecnico.ulisboa.pt

André Araújo
Instituto Superior Técnico
Lisbon, Portugal
andrejfaraujo@tecnico.ulisboa.pt

Miguel Neves
Instituto Superior Técnico
Lisbon, Portugal
miguelscfneves@tecnico.ulisboa.pt

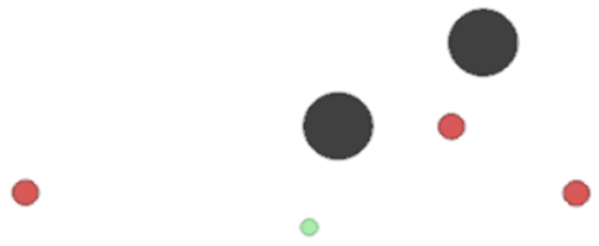

Figure 1: Example of an environment with one Prey (green circle), three Predators (red circles) and two obstacles (black circles).

ABSTRACT

This work presents a Deep Q Learning approach for developing controllers for multi-agent systems and compares it to other conventional solutions. We used a well-known environment - the 2D tag game - that incorporates multiple Predators chasing after a single faster Prey. With this environment we are able to asses what collaborative and competitive behaviour emerges from these approaches. We developed hardcoded controllers for both Prey and Predators and used them as the Baseline benchmark to compare the machine learning controllers with.

Although we expected the Advanced models to perform better than than the benchmark for both Prey and Predators, the Advanced Prey controller held worse results compared to the benchmark. We only saw effective collaborative behaviour from the Predators' Advanced models.

1 INTRODUCTION

Predator-Prey environments are of particular interest due to their relevance in the real world, be it to study and simulate Predator and Prey behaviour or to study how the environmental context affects the decision making and strategies of said agents. The game of tag allows for both aspects to be studied at the same time.

A traditional game of tag is quite simple: there is usually one Predator (commonly referred to as catcher) and several Prey (commonly referred to as runners), and the objective of the game is for the Predator to catch one of the Prey. Due to its simplicity, and the fact that cooperation is not encouraged in this setting, this system is not of great interest. There are only two possible outcomes: The Predator is faster than one of the Prey and it trivially catches it, or the Predator is slower than any one of the Prey and thus fails to catch anything.

To counter these limitations and create a more interesting environment, some modifications were made. Firstly, the simulated world was created with multiple Predators and only one Prey - which is faster than the Predators. This arrangement compels Predators to collaborate to catch the Prey. Secondly, obstacles were added making it possible to obtain randomized environments and study how the agents' behaviour and their strategies adapt to the environment, if at all.

In this project we set out to develop agents that fulfill their respective tasks successfully - catch the Prey or never get caught by the Predators. We are also interested in studying different approaches to implement the agents, and what behaviour patterns and strategies arise from these approaches.

2 ENVIRONMENT

The world is a 2D plane and there are 3 types of entities: a green agent, a team of red agents and obstacles. All entities are displayed in figure 1 and all agents are able to move in all directions with simulated inertia, drag and collisions.

The green agent - hereinafter referred to as "Prey" - is faster and receives a negative reward for being hit by adversaries (-10 for each collision). To stop the Prey from running to infinity, it is also penalized for exiting the center area. Assuming x is the distance to the center, the penalty the Prey receives follows the function:

$$penalty = \begin{cases} 0 & \text{if } x < 0.9 \\ (x - 0.9) * 10 & \text{if } x < 1 \\ min(e^{2x-2}, 10) & \text{if } x \geq 1 \end{cases} \tag{1}$$

The red agents - hereinafter referred to as “Predators”, are slower and are rewarded for hitting green agents (+10 for each collision).

All agents receive the same sensorial input, it includes

- agent’s self velocity and position,
- obstacles’ relative positions,
- other agents’ relative speed and positions.

Obstacles (large black circles) block the way. By default, there is one good agent, three adversaries and two obstacles.

Finally, the environment will run for a limited number of cycles, setting a time limit for the Predators to succeed.

3 ARCHITECTURE

The Prey and Predator controllers are completely independent as this is a competitive environment. Both Predators and Prey have a Baseline behaviour that the Advanced approach will compete against.

3.1 Baseline Behaviour

The Baseline behaviour is hardcoded and purely reactive and thus must output reproducible behaviour. In the following subsections we explain the implemented algorithms.

3.1.1 Prey. The Prey creates a polar graph representing the best directions to travel to. When a Predator is detected, the direction of the Predator is discounted by a Gaussian distribution with a sigma directly proportional to the relative distance of the Predator. Obstacles have the a similar effect although they discount lower values. All directions pointing outwards from the center are discounted proportionally to the distance from the center following a function similar to (1) in order to encourage the Prey to stay in bounds.

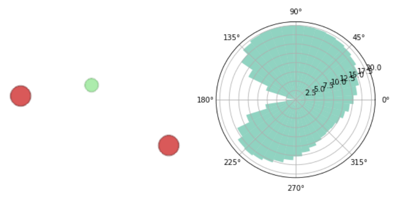

Figure 2: Example of a game state and the respective polar graph generated by the Prey.

3.1.2 Predators. The Predators will all move towards the Prey, mindless of obstacles they may encounter and lacking any form of cooperation.

3.2 Advanced Behaviour

Our Advanced approach utilizes Reinforcement Learning, more specifically Deep Q-Learning, to teach the Predators and the Prey how to achieve their goals.

3.2.1 Q-Learning. Q learning is a model free reinforcement learning algorithm that learns the value of executing a certain action in a particular state. These values, known as q-values, are iteratively updated by applying the Bellman equation:

$$Q(s_t, a_t) = Q(s_t, a_t) + \alpha(r_t + \gamma \max_a Q(s_{t+1}, a) - Q(s_t, a_t)) \quad (2)$$

The q-values are usually stored in a lookup table, known as q-table. An agent determines the action to take in a particular state by consulting the q-table and choosing the action with the highest q-value for that particular state.

3.2.2 Deep Q-Learning. Due to the continuous nature of the state space, the only way to use a q-table would be to discretize the state space. The challenge with doing this is knowing the necessary level of discretization, and fully understanding the error incurred by discretizing the state space. A far better approach is to use Deep Q-Learning [Mnih et al. 2013], where a neural network takes the environment state as input and is trained to output an approximation of the q-values of all the possible actions.

In Deep Q-Learning there are two distinct networks: the target network and the decision network. The decision network takes as input the state observed by the agent and outputs the optimal action according to the approximated q-values. The target network is sporadically updated with the weights and biases of the decision network, and is mainly used to provide stability. Every (state, action, new sate, reward) tuple is recorded in a replay buffer that acts as the memory of the agent. Back-propagation on the decision network is done by taking a sample from the replay buffer and using the target network to obtain an approximation of future q-values. The Bellman equation is then calculated with this approximation. The loss between the new q-values and the ones predicted by the decision network is computed and the gradient is applied.

3.2.3 Implementation. Each Prey and Predator was implemented with a feed forward neural network with an input layer, two hidden layers and an output layer, all densely connected. The input layer has a number of units equal to the size of the agent’s observation, the two hidden layers have sixty four units each, and the output layer has a number of units equal to the number of actions (five in our environment). All layers but the last have a Rectified Linear Unit (ReLU) activation function [Nair and Hinton 2010], with the last layer having a linear activation function. The Adam optimizer [Kingma and Ba 2014] was used as the optimization algorithm. We also used the Huber loss function [Huber 1964].

3.2.4 Training. The Advanced Agents were trained by playing against each other on an environment with one Prey, three Predators and two obstacles. This configuration was used because it provided the best ratio between the Prey being able to escape the Predators and the Predators being able to catch the Prey. The number of obstacles was kept small, while still providing the environment the needed variability, in order to not unnecessarily increase the model’s complexity. The agents were trained for 5 000 games, each one with 100 steps, for a total of 500 000 steps. We used a learning rate of 0.001 and a discount factor of 0.95. The agents explored the environment using an epsilon-greedy strategy starting at 1.0 and linearly decreasing to a minimum of 0.1 over 200 000 steps. The decision network was updated every 10 steps with a sample of size 1 024 taken from the replay buffer. The target network was updated every 1 000 steps.

Prey	Predator	Prey distance to center	Distance Predator-Prey	Steps until first capture	Captures
Baseline	Baseline	0.9788	0.9352	24.9200	0.390
Baseline	Advanced	1.0247	0.9964	38.8750	3.010
Advanced	Baseline	0.8059	0.7856	20.4167	8.410
Advanced	Advanced	0.8470	1.0316	23.6500	12.84

Table 1: Metrics of multiple model configurations aggregated by averaging on a game basis.

4 EVALUATION

As Prey and Predators have opposite goals, their approaches were evaluated and ranked separately. To evaluate the Predators' behaviour, their controller ran and competed against the Baseline behaviour of the Prey. Likewise, the Prey's approach was evaluated against the Baseline behaviours of the Predators. In a later stage, both the Prey and Predators competed using their Reinforcement Learning based approaches to see if cooperative and competitive strategies were still effective and assess if the algorithms handle a change in the adversary's behaviour.

4.1 Primary Metrics

These metrics are quantitative and have the purpose of providing an objective evaluation of the different approaches.

4.1.1 *Prey.*

- `Distance to Predators`
- `Steps until first capture`
- `Number of captures`
- `Distance to center`: the Prey is encouraged to stay close to the center

4.1.2 *Predator.*

- `Distance to Prey`
- `Steps until first capture`
- `Numbers of captures`: by the agent itself or other agents in the Predator team

4.2 Secondary Metrics

These metrics are qualitative and will serve to compare approaches with similar primary metric results and evaluate strategies.

These include movement patterns, jitters vs smooth movement, adaptability to obstacles, emergent strategies and their efficacy and signs of cooperation between Predators.

5 RESULTS

The experiment results are resumed in Table 1. Since the Advanced versions were trained on an environment with one Prey, three Predators and two obstacles, we used this configuration for the experiment environment. The data was obtained from a sample of 100 games with 100 steps each, and the metrics were computed by averaging the results on a game basis.

The Prey's Baseline behaviour is quite effective as the algorithm tries to maximize the distance between itself and all Predators whilst maintaining a proximity with the center of the map as to not receive an overwhelming negative reward. This approach is purely reactive but due to careful calibration of the algorithm, it proves to be more effective at reducing the number of collisions than the Advanced Prey controller.

Prey	Predator	Average reward
Baseline	Baseline	-0.4513
Baseline	Advanced	-0.8336
Advanced	Baseline	-1.3999
Advanced	Advanced	-1.0905

Table 2: Average reward Prey received per game step in multiple controller configurations

When taking into consideration the average distance from the center, the Advanced Prey has much better results, never going above the negative reward threshold. The massive difference in steps until first capture when matching the Baseline Prey or the Advanced Prey with the Advanced Predators is partially explained by this - most of the times the Predators would wait for the Baseline Prey to come closer to the center, as it was the expected behaviour of the Advanced Prey.

The Prey's Advanced controller developed a jitter technique that proved to be effective against the Predator's Advanced controller but ultimately damaged the Prey's reward when competing against the Predators' Baseline behaviour, as is displayed in the last two rows of Table 2.

5.1 Experiments

In this section we present a thorough analysis of the experiments realized.

5.1.1 Baseline Prey and Baseline Predators. The Prey's Baseline behaviour was calibrated to be optimal in this scenario and thus performed admirably well. As Predators move mindlessly towards the Prey, the Prey runs in circles around the center with the Predators trailing behind as seen in figure 3.

This is the optimal strategy for the Prey as it keeps Predators far whilst keeping the Prey close enough to the center as to not receive large negative rewards. The Prey still receives some negative rewards as the rules of the environment require the Prey to be very close to the center. This strategy represents a healthy trade off between distancing itself from the center and having enough room to evade Predators.

Sometimes, due to the existence of randomly placed obstacles in the environment, a Predator might get stuck behind one while chasing the Prey which causes an erratic trajectory for the Predator. This sometimes works in its advantage as it disrupts the Prey's otherwise stable strategy of running in circles.

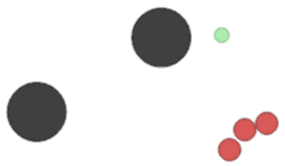

Figure 3: Example of the Baseline Prey running in circles with the Predators trailing behind from a safe distance.

From watching both Baseline behaviours we conclude that the Prey's Baseline is more effective than the Predators'. The Predators, as expected, show no signs of cooperation and sometimes collide with each other while blindly chasing the Prey.

5.1.2 *Baseline Prey and Advanced Predators.* From watching the Advanced Predators chasing the Prey we notice a clear improvement from their Baseline behaviour. The Predators spread out as to increase their reach and although the Prey is faster, its reactive nature and lack of planning leads to it often being trapped behind obstacles as Predators surround it. In figure 4 we can see the spreading technique in motion as the Prey gets closer to the left "wall".

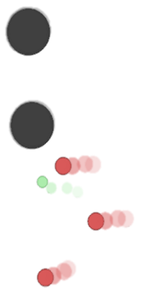

Figure 4: A display of how the Advanced Predators spread to cover more area.

Even when chasing the Prey the Predators tend to leave one or more agents behind as to cover more area as seen in figure 5. In the event that the Prey uses it speed to surround the leading Predators, the Predator(s) left behind will quickly catch up to it.

We can conclude by watching the Advanced Predators operate that they are vastly more effective at catching the Prey than their

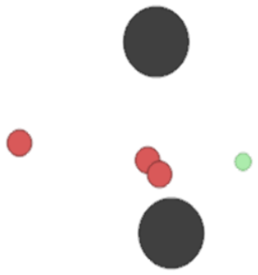

Figure 5: Advanced Predator controllers leaving one of the Predators behind as part of their strategy.

Baseline counterpart, reflected in table 1, where it increases by a factor of 10 the average number of collisions.

In the table we can also notice an increase on number of steps until first capture. This happens because most of Predator-Prey collisions that happened in the first configuration were due to the initial arrangement of the agents as the Predators would spawn either touching or very close to touching the Prey. This increase in number of steps until first capture indicates that the new Predator-Prey collisions that happen when the Advanced Predators compete against the Baseline Prey controller are due to planning and strategic movement and not random chance of initial agent placing.

5.1.3 *Advanced Prey and Baseline Predators.* The performance of the Advanced Prey versus the Baseline Predators perfectly demonstrates the achievements and the downfalls of the trained model.

The Prey displays how well it can evade the Predators, sometimes by waiting for them to concentrate and dodging at the last second (Figure 6) and sometimes by moving around the obstacles to block the path of the Predators (Figure 7).

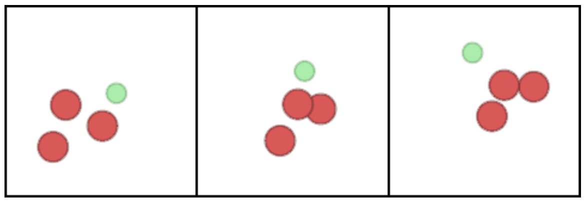

Figure 6: Example of the Advanced Prey evading the Baseline Predators.

However, since the Advanced Prey and the Advanced Predator were trained simultaneously, some overfitting can be seen against the Baseline Predators, resulting in slightly worse results in some metrics, as seen in the Table 1, for example the shorter time before the first capture.

When observing the agents in play it is apparent the behaviour that causes this reduction in performance, the Prey waits too much

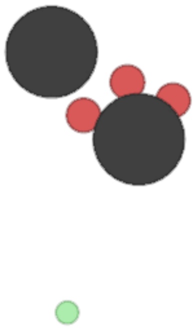

Figure 7: Example of the Advanced Prey using the obstacles to block the path of the Baseline Predators.

before sidestepping the Predators, probably expecting the Predators to encircle it as the Advanced Predator usually does, but the Baseline Predators simply move directly towards the Prey causing it to be caught in a corner (Figure 8).

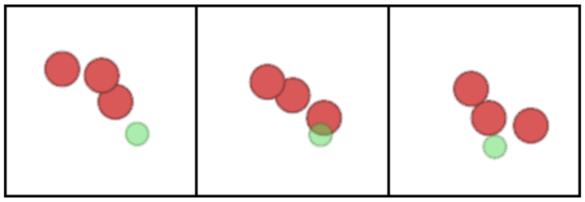

Figure 8: Example of the Advanced Prey sidestepping the Baseline Predators too late.

To fix this issue we could perhaps use the Baseline Predators as a validation set after each training iteration and graph its rewards compared to the training reward in order to identify when overfitting begins to occur and stop the learning earlier.

5.1.4 Advanced Prey and Advanced Predators. The most interesting experiment is of course the one where both Advanced Agents face each other, and it is here that the most exciting behaviour surfaces.

The Advanced Prey, as it does against the Baseline Predators, uses the obstacles to block the path of the Predators but the Advanced Predators have learned to cooperate by splitting up and traversing the obstacles through different sides in order to trap the Prey behind it (Figure 9).

In general the Predators seemed to have learned that the best overall strategy is to encircle the Prey reducing it's escape routes and pretty much guaranteeing at least one catch when it tries the escape their barricade.

The Prey exhibits evasive maneuvers like the simple side-step when the Predators come toward it in a corner, but more interestingly, it learned to use the physics of the environment to it's advantage: sometimes it runs straight into an obstacle in order

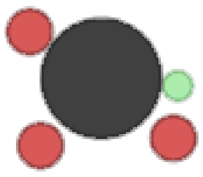

Figure 9: Example of the Advanced Predators trapping the Advanced Prey behind an obstacle.

to bounce off of them while maintaining momentum and leaving behind the Predators (Figure 10).

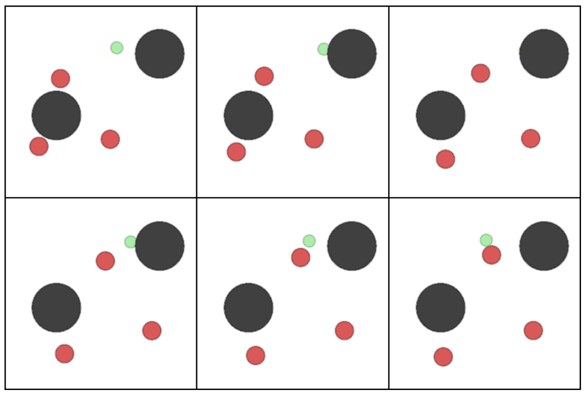

Figure 10: Example of the Advanced Prey bouncing of an obstacle to lose the Predators.

5.2 Final Experiment Remarks

Overall we can see that the Baseline Prey developed is very efficient in avoiding both the Baseline and Advanced Predators by maintaining its distance and running in circles near the center of the environment. The trained Advanced Prey, while displaying an impressive adaptability to the environment by using it both as cover and to quickly switch its direction, is not as efficient as the Baseline Prey it terms of avoiding the Predators.

Meanwhile the Advanced Predators show great improvements over the Baseline against both Prey models, displaying great use of the obstacles, trapping Preys behind them, and impressive cooperation skills by blocking escapes and encircling them.

Future experiments could try to use the Baselines as validation when training the Deep Q Networks to avoid overfitted behaviour.

6 CONCLUSION

The Baseline Prey developed was a clever solution and worked better than hoped, being one of the highlights of the project.

The Deep Q Network also proved to be a good model for both prey and predators, developing very interesting behaviour and showing it is a good solution for the presented problem. Perhaps with more training and using some techniques to avoid overfitting, the Deep Q Network based Prey can prove to be even better than the Baseline Prey.

Even with some drops in performance, with these models we managed to accomplish our goal of training and identifying successful cooperative strategies and approaches to overcome situations where a Prey is outnumbered and how the environment affects these behaviours.

REFERENCES

Peter J. Huber. 1964. Robust estimation of a location parameter. *Annals of Mathematical Statistics* 35, 1 (March 1964), 73–101. https://doi.org/10.1214/aoms/1177703732

Diederik P. Kingma and Jimmy Ba. 2014. Adam: A Method for Stochastic Optimization. http://arxiv.org/abs/1412.6980

Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Alex Graves, Ioannis Antonoglou, Daan Wierstra, and Martin Riedmiller. 2013. Playing Atari with Deep Reinforcement Learning. (2013). http://arxiv.org/abs/1312.5602

Vinod Nair and Geoffrey E. Hinton. 2010. Rectified Linear Units Improve Restricted Boltzmann Machines.. In *ICML*, Johannes Fürnkranz and Thorsten Joachims (Eds.). Omnipress, 807–814. http://dblp.uni-trier.de/db/conf/icml/icml2010.html#NairH10