Towards Ecosystem Management from Greedy Reinforcement Learning in a Predator-Prey Setting

Fabian Ritz, Felix Hohnstein, Robert Müller, Thomy Phan, Thomas Gabor, Carsten Hahn
and Claudia Linnhoff-Popien

Mobile and Distributed Systems Group, LMU Munich, Germany
fabian.ritz@ifi.lmu.de

Abstract

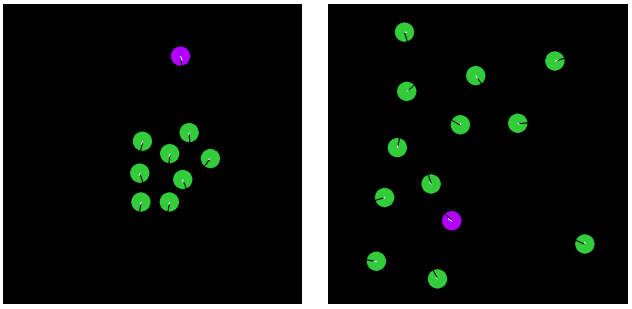
This paper applies reinforcement learning to train a predator to hunt multiple prey, which are able to reproduce, in a 2D simulation. It is shown that, using methods of curriculum learning, long-term reward discounting and stacked observations, a reinforcement-learning-based predator can achieve an economic strategy: Only hunt when there is still prey left to reproduce in order to maintain the population. Hence, purely selfish goals are sufficient to motivate a reinforcement learning agent for long-term planning and keeping a certain balance with its environment by not depleting its resources. While a comparably simple reinforcement learning algorithm achieves such behavior in the present scenario, providing a suitable amount of past and predictive information turns out to be crucial for the training success.

Introduction

Co-evolving ecosystems in nature often strive for an equilibrium between the involved parties, see Rosenzweig and MacArthur (1963). In the present paper, a predator-prey interaction is considered, where for example a shark may hunt smaller fish at roughly the rate that the fish can compensate via reproduction. In nature, it is assumed that an equilibrium state is approached through many generations of evolution and none of the involved individuals follows any goal but its own self-interest: To greedily eat as many fish as you can, in case of the shark. However, as not extinguishing the fish altogether is obviously a superior strategy, it should follow from the shark's pure self-interest to spare a few fish in order to eat their progeny later.

While such foresight is rarely observed in nature where equilibria are usually approached 'blindly' by counteracting self-interest, we suggest that human-made systems might be able to approach a state of balance deliberately. For this paper, the following question is given: Under which circumstances can a purely self-interested predator learn individually to spare prey for later benefit? In general, reinforcement learning (RL) should be able to find an economical strategy for long-term benefit. In practice however, rewards in this setting are sparse and initially deceptive which might be prohibitive for learning such far-sighted strategies. Therefore, the training requires meticulous configuration of the horizon of past observations and the discount factor of future rewards, which are discussed later. Of course, the optimal strategy also changes with domain parameters like the prey's reproduction rate. It is shown that two-stage learning, i.e. first learning the purely greedy objective and then generalizing it to the non-greedy setting, is able to yield effective results in the present domain.

This paper can be regarded as a first step to a guideline on how to develop intelligent agents. Even without special tools or goal functions, these agents actively sustain their continuous reward in an open-ended domain rather than just maximizing reward within a short time frame, thus combining an artificial life simulation with one of the most challenging problems in RL, see Stout et al. (2005).



(a) Predator and swarming prey (b) Predator and turn-away prey

Figure 1: Visualization of the continuous, two dimensional predator-prey environment. The predator agent is colored purple, the prey agents are colored green.

Foundations

Reinforcement Learning

Similar to Kaelbling et al. (1998), the problem is formulated as a *Partially Observable Markov Decision Process* $M = \langle \mathcal{S}, \mathcal{A}, \mathcal{P}, \mathcal{R}, \mathcal{O}, \Omega, b_0 \rangle$, where $\mathcal{S}$ is a set of states, $\mathcal{A}$ is the set of actions, $\mathcal{P}(s_{t+1}|s_t, a_t)$ is the transition probability, $\mathcal{R}(s_t, a_t)$ is the scalar reward, $\mathcal{O}$ is a set of observations,

$\Omega(o_{t+1}|s_{t+1}, a_t)$ is the observation probability and b_0 is a probability distribution over initial states $s_0 \in \mathcal{S}$. It is always assumed that $s_t, s_{t+1} \in \mathcal{S}$, $a_t \in \mathcal{A}$, and $o_t, o_{t+1} \in \mathcal{O}$ at time step t. A *history* $\tau_t = \left[a_0, o_1, a_1, o_2, ..., a_{t-1}, o_t\right]$ is a sequence of actions and observations. As in Mnih et al. (2015), only histories with a fixed length of h are regarded, where old entries are successively replaced by new ones.

The goal is to find a *policy* $\pi(\tau_t) \in \mathcal{A}$ which maximizes its value function $Q^\pi(\tau_t, a_t) = \mathbf{E}[\sum_{k=0}^{\infty} \gamma^k \cdot \mathcal{R}(s_{t+k}, a_{t+k})|\tau_t, a_t]$ for each history τ_t and each action a_t. $\gamma \in [0, 1]$ is the discount factor. An *optimal policy* π^* has a value function $Q^{\pi^*} = Q^*$ with $Q^*(\tau_t, a_t) \geq Q^{\pi'}(\tau_t, a_t)$ for all τ_t, a_t, and $\pi' \neq \pi^*$.

π^* can be approximated with *reinforcement learning (RL)*, where π^* is learned with experience tuples $\langle o_t, a_t, r_t, o_{t+1} \rangle$, which are obtained from agent interaction with the environment. *Q-Learning* is a popular approach to RL, where $\hat{Q} \approx Q^*$ is approximated with the following update rule (Watkins and Dayan (1992)):

$$\hat{Q}(\tau_t, a_t) \leftarrow (1 - \alpha)\hat{Q}(\tau_t, a_t) + \alpha(y - \hat{Q}(\tau_t, a_t)) \quad (1)$$

where $y = r_t + \gamma \cdot max_{a_{t+1}} \hat{Q}(\tau_{t+1}, a_{t+1})$ and $\alpha \in [0, 1]$ is the learning rate. Following Mnih et al. (2015) and Hausknecht and Stone (2015), current state-of-the-art RL in POMDPs is implemented with deep learning using, e.g. *Deep Q-Networks (DQN)*.

Swarm Behavior

Flocking or swarm behavior is a widely observed phenomenon in nature. Although the entities might have self-interested goals like evading predators, they may group themselves together, e.g. to decrease the overall flow resistance or to gain more information as collaborative observation may be superior to the observation of a single individual. A fundamental work of Reynolds (1987) on swarm simulation formulates three basic behavior rules for autonomously acting units (*Boids*) to form a swarm. *Cohesion* defines how to navigate to the centered position of the neighboring Boids, *Separation* defines how to keep a minimum distance from other Boids and *Alignment* defines how to adjust the own alignment to that of the neighboring Boids. These rules are treated as weighted forces and are based solely on local information. Each Boid requires only position and the movement direction of its nearest neighbors but does not need an overview of the entire swarm. Subsequently, Reynolds (1999) developed algorithms to produce natural behavior in situations where individuals escape or pursue a target. This is achieved by combining *separation* as described above and the so-called *seek* or *flee* behavior. In essence, a continuous force is applied between a Boid's current and its target's position. The mathematical sign determines whether it attracts (*seek*) or deflects (*flee*) the Boid's direction of movement.

Swarm behavior emerging in predator-prey scenarios in presence of self-interested agents has also been observed in Multi-Agent reinforcement learning (MARL) by Morihiro et al. (2008) who shaped rewards according to the three Boid rules. This indicates that swarming may be an optimal prey strategy in presence of a predator. Additionally, Hahn et al. (2019) investigated whether MARL can achieve similar results in a continuous environment without explicitly rewarding a certain distance to neighbors. Regarding survival time, the learned policies performed better than acting strictly according to the three Boid rules. However, these policies did not consistently beat the turn-away strategy, a flee strategy completely ignoring swarming. Subsequently, Hahn et al. (2020) argued that this resulted from transferring the policies into scaled up scenario. They moreover showed empirically that in their scenario, staying in a swarm is a Nash Equilibrium in terms of survival time. Also, Olson et al. (2016) reported that the emerging prey behavior strongly depends on the scenario. Therefore, the two most contrary prey survival strategies are utilized in this paper. The prey agents either flee from the predator while maintaining a swarm formation, referred to as *swarming* agents, or flee individually while ignoring cohesion and alignment, referred to as *turn-away* agents, though still respecting separation to not collide with other prey agents.

Related Work

While RL gained widespread popularity in recent years, its application to swarms and the respective Multi-Agent Systems has received considerably less attention, see Khan et al. (2018). Hüttenrauch et al. (2017) proposed different actor-critic architectures for MARL scenarios where the actor only has access to a single agent's local observation while the critic has access to the entire state of the world. Their agents had to coordinately solve complex tasks in a two-dimensional physics environment. Technically similar, Lowe et al. (2017) analyzed actor critic approaches in predator-prey scenarios with predator and prey as learning entities. Subsequently, Hüttenrauch et al. (2019) investigated how to efficiently represent an agent's local observation in an environment with many homogeneous agents in a pursuit-evasion scenario. The authors propose to treat each agent as a sample of a distribution and use the empirical mean embedding as input for a decentralized policy, yielding a representation of invariant size with respect to the number of visible agents. Yang et al. (2018) investigated the dynamics of large predator prey populations in a grid world trained with modified DQN. In their scenario, predators could die from starvation. They observed a wax-and-wane shape between predator populations, which learned to hunt efficiently, consequently shrinking the prey population on the short term. On the long term, the predator population shrinked due to starvation, leading to a rise the prey population again. These population dynamics are consistent with the Lotka-Volterra model

proposed by Lotka (1956). A different aspect of swarming was introduced by Pinsler et al. (2018). The authors use *Inverse RL* to recover the underlying reward function of bird flocking behavior in absence of predators. This enables the reproduction of flocking behavior through RL. Additionally, the reward functions are used to learn a leader-follower hierarchy. Later, Hahn et al. (2019) showed that swarm behavior can solely emerge from self-interested agents that try to avoid being caught by a predator. In their scenario, the relatively small environment wrapped around at the edges, i.e. agents leaving to the left immediately re-enter from the right. Subsequently, Hahn et al. (2020) showed empirically that in this scenario, swarming behavior may form a Nash equilibrium and an individual fleeing behavior would improve the prey populations survival. Yet, single agents may not have an incentive to leave the swarm as this would turn them into an easier target in free space compared to agents remaining within the swarm formation. While their scenario is similar to that of the present paper, their prey agents were trained with RL and their predator followed a static policy.

In RL, it is often important to not only concentrate on the immediate reward but to act far sighted. This becomes more apparent when the agent has to plan ahead for many time steps in order to achieve its goal. Prematurely focusing on rewards in the near future might cause the agent to get stuck in local optima without actually reaching the desired goal at all, see Reddy et al. (2019). Consequently, an architecture has to be designed such that information can be retained over a large number of time steps. Jaderberg et al. (2019) train agents in a capture-the-flag 3D multiplayer game to operate on two timescales. A fast Recurrent Neural Network (RNN) models the quickly changing temporal dynamics of the environment while a slow RNN accounts for temporal correlations and promotes memory. This approach allows the agent to develop long-term strategies. Vinyals et al. (2019) use RL to master StarCraft 2, one of the most challenging real-time strategy games. As a game may take up to an hour, the ability of long-term planning is essential. Actions taken at early stages of the game may significantly influence the outcome. Moreover, their effect is not measurable immediately and it can take a long time until they pay off. The authors combine various neural network architectures, e.g. *Pointer Networks* developed by Vinyals et al. (2015), *LSTMs* developed by Hochreiter and Schmidhuber (1997) and *Transformers* developed by Vaswani et al. (2017), to enable long-term sequence modeling and propose a game-theoretic, population-based training curriculum. To increase sample efficiency, Hafner et al. (2019b), Hafner et al. (2019a) and Ha and Schmidhuber (2018) use an RNN to explicitly learn the environment's dynamics. This enables the agent to "dream" and plan ahead in its own version of the environment.

However, in the present work, the predator gains the ability to memorize and plan ahead through the concatenation of an adjustable number of past states in conjunction with the RL discount factor γ, which weights the influence of future rewards. In the present setting, this it sufficient to achieve an adaptive, far sighted behavior. Therefore, more sophisticated, parameter heavy and hard-to-tune architectures can be avoided. Inspired by Vinyals et al. (2019), a multi-step training process is employed. However, the present pipeline is hand-crafted and considerably less complex.

In non-cooperative game theory, it is assumed that agents act self-interested and independently. In the case of shared resources, this often leads to the tragedy of the commons as reported by Lloyd (1833)), where resources are exhausted through selfish behavior instead of being shared fairly. Moreover, non-cooperative game theory does not support the discovery of socially positive equilibria and is hard to adopt to complex environments. Perolat et al. (2017) study common-pool resource appropriation problems through the lens of MARL to observe the emergent behavior of independent, self-interested learners. Instead of specifying the strategy, e.g. tit-for-tat, this approach allows the agents to learn the strategy themselves. They report the emergence of different strategies with the parameters for the environment being changed and measure them using social metrics such as peace, efficiency, equality and sustainability. Even though the present paper only considers single-agent RL, it also finds sustainable resource management emerging through pure self-interest.

Domain

Environment

All agents are defined as unicycles, a commonly used agent model in mobile robotics, featuring a two-dimensional position, linear velocity (*speed*) and angular velocity (*direction*). The agents cannot access speed and direction directly but add the respective changes as accelerating or decelerating forces. For example, to change direction, the agent needs to adjust his *orientation* and accelerate. Per simulation step, each agent can either reproduce or adjust its speed and orientation by a certain amount. The maximum speed is determined by the ratio of the agent's linear acceleration divided by the respective friction constant. Technically, this results in a simulation with double integrator dynamics. The continuous, two-dimensional state space is bounded, the limits act as walls. Agents can collide with walls and other agents. If one of the agents in a collision is the predator, the other agent gets removed from the simulation. In any other case, an elastic collision is performed with an elasticity constant determining how much kinetic energy is preserved. In the context of collisions, all agents have the same weight. The most important constants are summarized in Table 1. A visualization of the environment is depicted in Fig.1.

Environment size	30×30
Diameter of predator and prey	2.00
Max number of prey	10.0
Radius of prey's perception	10.0
Max observed prey	3.00
Max acceleration of predator	0.06
Max acceleration of prey	0.04
Max orientation change of predator	$\frac{\pi}{3}$
Max orientation change of prey	π
Prey reproduction after n steps	100
Friction of predator	0.10
Friction of prey	0.08
Elasticity constant	0.20

Table 1: Key domain parameters.

Actions

The non-RL agents can access the full, continuous action space, meaning they can change their orientation by any value between $-\pi$ and π and accelerate by any value between -1 and 1 as displayed in Tab. 1. However, the non-RL agents were adjusted to always choose the maximum linear acceleration and only control their orientation according to the respective strategy. As DQN is only capable of choosing between discrete actions, the continuous action space is divided into six actions for the RL agent: Five actions combine full linear acceleration with orientation changes of $-\frac{\pi}{3}$, $\frac{\pi}{6}$, 0, $\frac{\pi}{6}$, $\frac{\pi}{3}$ and NO-OP without any acceleration as the sixth action. Since the predator may not reproduce in the present setting, reproduction is a unique feature of prey agents. If a prey is not caught within a certain amount of steps and no predator is inside its perception radius, it will spawn an identical copy of itself right beside its current position.

Observations

While the predator agent has an unlimited observation distance, prey agents receive information about walls, predators and other prey only within their local neighborhood. The neighborhood is defined as a circle centered at the respective agent's centroid with a certain radius. Predator and prey agents can sense up to two walls, three predator agents (yet there is only one predator in the present scenario) and three prey agents. If more entities are visible, they are discarded. All information is provided by an ordered vector with constant length and fixed entity offsets as seen from the respective agent. For example, walls are always placed upfront in the observation vector. If no wall is visible, zero padding preserves the offset of following entities, e.g. other agents. If multiple walls are perceivable, they are sorted by distance. While walls can be described solely with positional information (distance), other agents are additionally characterized by their current orientation. Whether and when other agents may reproduce is hidden. Distance and orientation are expressed via polar coordinates.

Training	
Training episodes	20000
Steps per episode	500
Replay buffer capacity	50000
Batch size	32
Steps until target net update	10000
Exploration	
Function	ϵ-greedy
Decay	linear, per step
Start value	1.0
End value	0.001
Neural network	
Hidden dense layer 1	32
Hidden dense layer 2	16
Activation function	ELU
Loss function	Huber
Optimizer	Adam
Max learning rate	0.0005

Table 2: Key DQN hyperparameters.

Reward

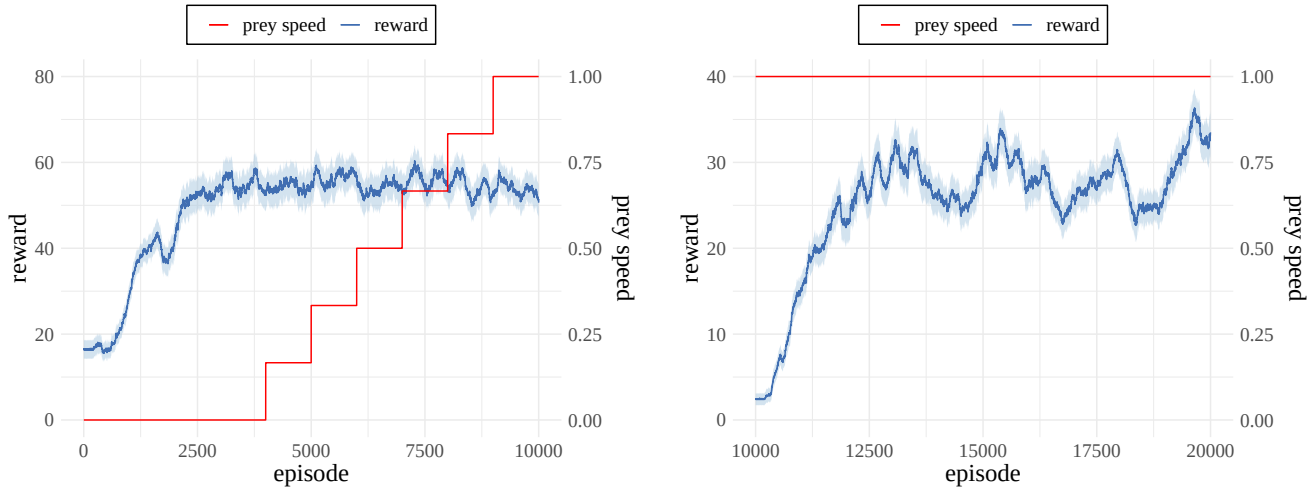
The predator agent receives a reward only for catching a prey agent. Every other state is classified as neutral, providing neither reward nor punishment, which ensures that no behavioral bias is introduced. In the present environment, the predator catching a prey is a comparably rare event, resulting in a sparse reward setting. During the experiments, a reward of $+10$ per catch yielded the best results in all training stages.

Experimental Setup

During prior experiments, the predator agent was observed to only develop advanced strategies with foresighted behavior once the basics of navigation and hunting had been learned. Inspired by curriculum learning, the training was split into two consecutive stages, see Fig. 2. In addition, the turn-away agents turned out to be more difficult to hunt, see Fig. 3. Therefore, only turn-away agents are used during training. In both stages, the predator agent was trained with DQN and stacked observations. The most important hyperparameters are listed in Tab. 2. RL discounts future rewards with a factor γ, which varied between 0.970 and 0.999 in the experiments. If not stated differently in the respective figure, γ is set to 0.990.

Two-Stage Training Process

During the first training stage, the predator agent shall learn to greedily hunt prey agents. Therefore, it is placed within the environment besides a number of prey agents whose movement is initially blocked, forming a dense reward setting. The number of spawned prey is chosen randomly between 1 and 10 and their reproduction is disabled. After the exploration phase, prey speed is partially increased every 1000 episodes. Initially facing non-moving targets, the



(a) Training stage I (dense reward setting)

(b) Training stage II (sparse reward setting)

Figure 2: Training stages of the predator agent. During the first training stage on the left, the predator agent learns to catch multiple non-reproducing prey agents. During the second training stage on the right, the predator learns to spare the initial prey agent until it reproduces to be able to catch more than one prey agent afterwards.

predator has to adapt to successively faster moving prey until the end of stage I.

During the second training stage, the predator agent shall learn to utilize his greedy behavior within a balanced, economic strategy with respect to the number of remaining prey agents. Therefore, the number of randomly spawned prey is capped at 3. Prey reproduction is enabled and prey speed remains uncapped. Yet, the second stage is parameterized such that the predator would be able to catch the initial prey before it reproduces if he desires to do so. Especially in the beginning of episodes, there is comparatively few prey, resulting in a more sparse reward setting. To maximize the overall reward, the optimal strategy would be to not hunt the prey until it reaches a stable population by reproduction. The most important detail of the second stage is that episodes do not end if all prey is caught. Effectively, this leaves the predator without any future rewards for the rest of this episode when being too greedy in the beginning.

Scenarios

To assess the RL agent's performance, several hand-crafted algorithms were implemented. The *static* predator uses a purely greedy heuristic that always chases the nearest prey agent regardless of the number of remaining prey. The *static-rand* predator either chases the nearest prey or chooses a random action, both with a probability of 50%. The *static-wait* predator uses an economic heuristic that only chases the nearest prey if there is at least one more prey agent within the environment.

Furthermore, a base scenario was created, from which all evaluations are derived. All evaluation scenarios use the same environment parameterization as the training scenario, which is listed in Tab. 1. The base scenario proceeds the second training stage. However, the predator agent spawns with only one initial prey agent to hunt. The prey can move at full speed and reproduce. This scenario evaluates whether the predator is able to spare prey in the beginning and hunt effectively later on.

In a first variation, the number of initial prey is increased stepwise from 1 to 10. The episode length is expanded to 1000 steps and prey reproduction is disabled, granting the predator enough time to catch all prey agents if he desires to do so. This scenario evaluates the impact of different amount of prey on the predator's behavior.

In a second variation, the number of steps until the initial, single prey agent reproduces is increased while the number of steps until the predator agent catches the first prey agent is tracked. This scenario evaluates whether a different reproduction time influences the predator's behavior.

Results

Using the base scenario, Fig. 3 depicts the performance of the strongest RL predator[1] against swarming and turn-away prey. After completing the first training stage, the RL predator's performance is on par with the static, greedy predator. After completing the second training stage, the RL predator outperforms all static predators regardless of the preys' survival strategy. While forcing the static predator to choose 50% of its actions randomly increases its perfor-

[1] A supplementary video of this agent hunting turn-away prey is available online: `https://youtu.be/RrqnaZ3CaxU`

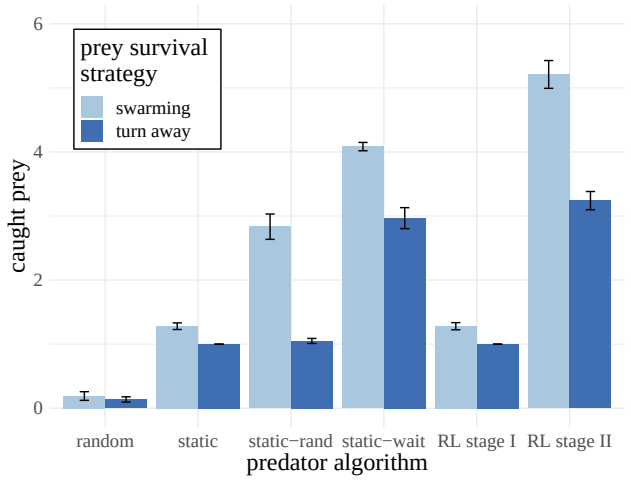


Figure 3: Average number of caught prey of different predators with swarming and non-swarming prey. This scenario starts with a single, reproducing prey agent. Average and 0.95 CI of 300 episodes are reported.

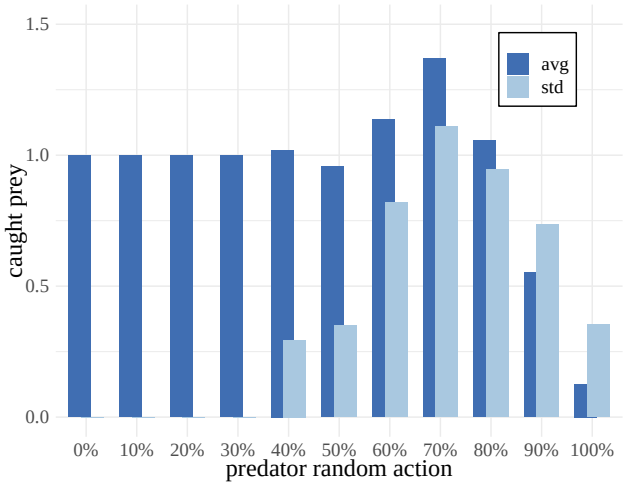


Figure 4: Effect of random actions on the static predator's average number of caught prey. This scenario starts with a single, reproducing prey agent. Average (avg) and standard deviation (std) of 300 episodes are reported.

mance against swarming prey, this neither achieves the performance of the RL predator, nor yields performance gains against turn-away prey. More specifically, Fig. 4 shows that 70% random actions are required to significantly increase the hunting success of the static predator against turn-away prey but come at the cost of very high variance. Using the first variation of the base scenario, Fig. 5 compares the number of spawned prey with the number of remaining prey at the end of an episode. Regardless of the initial number of prey, the RL predator always spares at least one prey agent. Using the second variation of the base scenario, Fig. 6 puts the number of steps until the prey reproduces in relation to the number of steps until the RL predator catches the first prey. The time until the predator catches the first prey increases with higher time until prey reproduction, indicating a linear correlation.

Using the base scenario, Fig. 7 shows which combination of past observations (trace length) and long term rewards (γ) results in the strongest RL predators. With $\gamma = 0.99$, a trace length of 20 results in approximately 2.4 caught prey per episode. Decreasing the trace length leads to a decrease of caught prey until 1 and increasing the trace length causes a sharp drop of caught prey towards 0. The same can be observed at $\gamma = 0.999$ with performance peaking at trace length 1, whereas at $\gamma = 0.97$ the performance peaks between 20 and 40. Further explanation is provided by the number of prey reproductions. At a low trace length, the prey rarely reproduces, indicating that the predator does immediately catch the first prey. With high trace length, prey reproductions peak at around 10, indicating that the predator does not catch any prey at all. Overall, the RL predator performs best at around 4 prey reproductions.

Discussion

In the basic evaluation scenario, only adaptive hunting strategies that keep a minimum distance to the prey for a certain amount of time (until it reproduces) lead to higher scores than 1. After the first training stage, the strongest RL predator performs similar to a (simple) greedy predator. After completing the second training stage, however, the RL predator is capable of outperforming all static, greedy predator algorithms regardless of the prey's survival strategy. The effect of turn-away prey being more difficult to hunt than swarming prey in a restricted environment is not surprising and was also reported in prior work of Hahn et al. (2019, 2020). While the *static-wait* predator does always spare exactly one prey agent, the RL predator reaching higher scores indicates that RL agents are able to surpass the economic capabilities of handcrafted heuristics. Further, the results demonstrate that adding random actions, which poses a chance of sparing prey from time to time, does not lead to comparable results.

Considering that the predator does not know whether prey agents can reproduce, the experiments with a varying initial number of prey and disabled prey reproduction clearly demonstrate that the RL predator hunts effectively but deliberately spares at least one prey agent, expecting the prey population to recreate. Further considering that the predator does not know when prey will reproduce, the experiments with the opposite scenario of one initial prey and prey reproduction after varying time further emphasize that the RL predator did not simply learn to wait a certain amount of steps until starting the hunt but effectively considers the actual size of the prey population.

While the results demonstrate that a comparably simple neu-

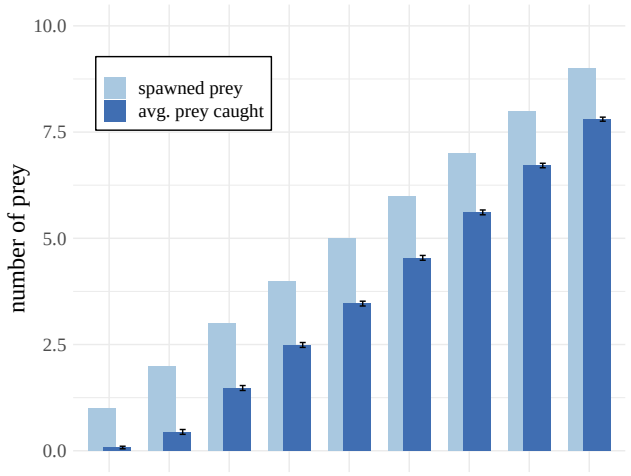


Figure 5: Number of prey spared by the strongest RL predator. The scenario starts with a fixed number of non-reproducing prey agents. Average and 0.95 CI of 300 episodes are reported.

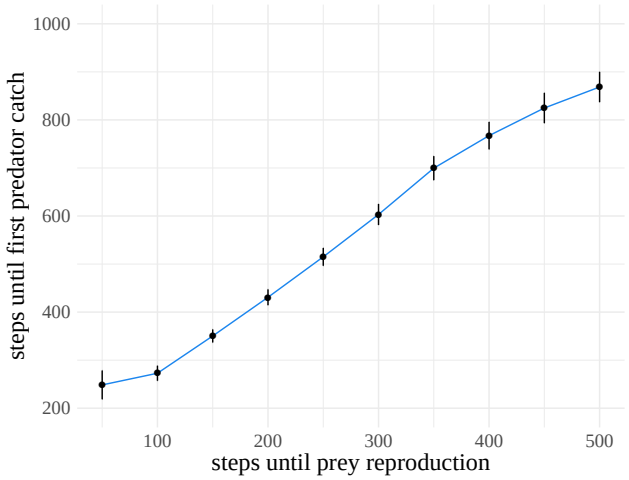


Figure 6: Correlation of the number of steps until the prey reproduces and the first catch of the strongest predator. The scenario starts with a single, reproducing prey agent. Average and 0.95 CI of 300 episodes are reported.

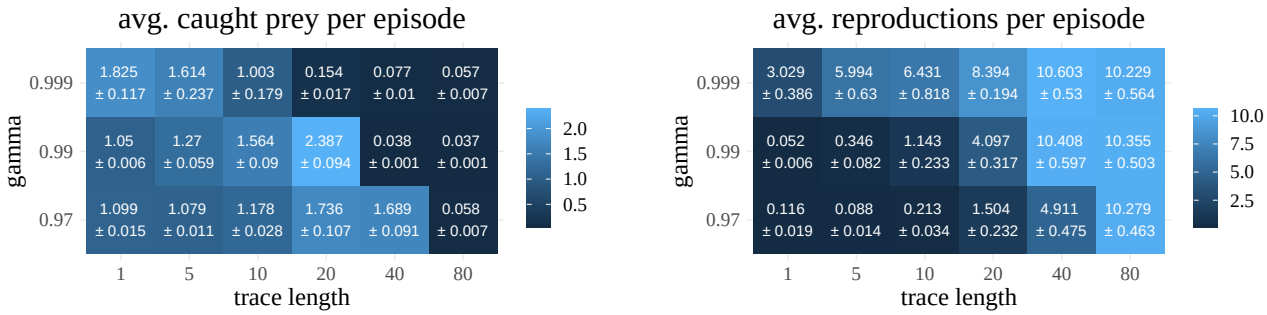


Figure 7: Impact of long term rewards (gamma) and past observations (trace length) on the training success (number of caught prey) of the RL predator. Per combination, 20 predator agents were trained with reproducing, non-swarming prey agents. The plot on the left reports the average prey catches and the 0.95 CI of the 10 strongest predator agents, the plot on the right contains the respective prey agents' reproductions.

ral network architecture is sufficient to achieve this behavior when trained with methods of curriculum learning, the impact of γ and trace length on the RL predator's performance is remarkable. This indicates that for a given neural network, there only is a narrow path between too little (small trace length) and too much information (high trace length).

Conclusion

This paper applied RL to train a predator to hunt multiple prey, which are able to reproduce, in a 2D simulation. It was shown that, using methods of curriculum learning, long-term reward discounting and stacked observations, an RL-based predator could achieve a economic strategy of hunting only if there is still prey left to reproduce in order to maintain the population. Consequently, purely selfish goals were be sufficient to motivate an RL agent for long-term planning and keeping a certain balance in the environment by not depleting its resources. Yet, the experiments also showed that learning a long-term optimal, sustainable behavior is a complex task that requires a certain amount of memory capacity (past observation length, future reward discounting) and maybe even brain plasticity (curriculum learning) to arise on an individual level out of self-interest. This coincides with such behavior being practically non-existent in nature. However, it is important to note that this paper neither considered the dynamics arising in presence of multiple predators, nor predator starvation, allowing for a line of future research. It is suspected that fully sustainable behavior cannot always be generated from self-interest only, but even then it is especially important to recognize which parts can and which parts need to be given as separate goals if we want the intelligent agent to manage its ecological surroundings.

References

Ha, D. and Schmidhuber, J. (2018). Recurrent world models facilitate policy evolution. In *Advances in Neural Information Processing Systems 31*, pages 2451–2463. Curran Associates, Inc.

Hafner, D., Lillicrap, T., Ba, J., and Norouzi, M. (2019a). Dream to control: Learning behaviors by latent imagination. *arXiv preprint arXiv:1912.01603*.

Hafner, D., Lillicrap, T., Fischer, I., Villegas, R., Ha, D., Lee, H., and Davidson, J. (2019b). Learning latent dynamics for planning from pixels. In *International Conference on Machine Learning*, pages 2555–2565.

Hahn, C., Phan, T., Feld, S., Roch, C., Ritz, F., Sedlmeier, A., Gabor, T., and Linnhoff-Popien, C. (2020). Nash equilibria in multi-agent swarms. In *12th International Conference on Agents and Artificial Intelligence (ICAART 2020)*.

Hahn, C., Phan, T., Gabor, T., Belzner, L., and Linnhoff-Popien, C. (2019). Emergent escape-based flocking behavior using multi-agent reinforcement learning. In *Conference on Artificial Life (ALIFE 2019)*.

Hausknecht, M. and Stone, P. (2015). Deep recurrent q-learning for partially observable mdps. In *2015 AAAI Fall Symposium Series*.

Hochreiter, S. and Schmidhuber, J. (1997). Long short-term memory. *Neural computation*, 9(8):1735–1780.

Hüttenrauch, M., Šošić, A., and Neumann, G. (2017). Guided deep reinforcement learning for swarm systems. *arXiv preprint arXiv:1709.06011*.

Hüttenrauch, M., Sosic, A., and Neumann, G. (2019). Deep reinforcement learning for swarm systems. *JMLR*, 20:54:1–54:31.

Jaderberg, M., Czarnecki, W. M., Dunning, I., Marris, L., Lever, G., Castaneda, A. G., Beattie, C., Rabinowitz, N. C., Morcos, A. S., Ruderman, A., et al. (2019). Human-level performance in 3d multiplayer games with population-based reinforcement learning. *Science*, 364(6443):859–865.

Kaelbling, L. P., Littman, M. L., and Cassandra, A. R. (1998). Planning and acting in partially observable stochastic domains. *Artificial intelligence*, 101(1-2):99–134.

Khan, M. M., Kasmarik, K., and Barlow, M. (2018). Toward computational motivation for multi-agent systems and swarms. *Frontiers in Robotics and AI*, 5:134.

Lloyd, W. F. (1833). *Two lectures on the checks to population*.

Lotka, A. J. (1956). Elements of physical biology. 1925. *Williams and Wilkins, Baltimore*.

Lowe, R., Wu, Y. I., Tamar, A., Harb, J., Abbeel, O. P., and Mordatch, I. (2017). Multi-agent actor-critic for mixed cooperative-competitive environments. In *Advances in neural information processing systems*, pages 6379–6390.

Mnih, V., Kavukcuoglu, K., Silver, D., Rusu, A. A., Veness, J., Bellemare, M. G., Graves, A., Riedmiller, M., Fidjeland, A. K., Ostrovski, G., et al. (2015). Human-level control through deep reinforcement learning. *Nature*, 518(7540):529–533.

Morihiro, K., Nishimura, H., Isokawa, T., and Matsui, N. (2008). Learning grouping and anti-predator behaviors for multi-agent systems. In *Knowledge-Based Intelligent Information and Engineering Systems*, pages 426–433, Berlin, Heidelberg.

Olson, R. S., Knoester, D. B., and Adami, C. (2016). Evolution of swarming behavior is shaped by how predators attack. *Artificial Life*, 22(3):299–318. PMID: 27139941.

Perolat, J., Leibo, J. Z., Zambaldi, V., Beattie, C., Tuyls, K., and Graepel, T. (2017). A multi-agent reinforcement learning model of common-pool resource appropriation. In *Advances in Neural Information Processing Systems*, pages 3643–3652.

Pinsler, R., Maag, M., Arenz, O., and Neumann, G. (2018). Inverse reinforcement learning of bird flocking behavior.

Reddy, S., Dragan, A. D., Levine, S., Legg, S., and Leike, J. (2019). Learning human objectives by evaluating hypothetical behavior. *arXiv preprint arXiv:1912.05652*.

Reynolds, C. W. (1987). Flocks, herds and schools: A distributed behavioral model. In *Proceedings of the 14th Annual Conference on Computer Graphics and Interactive Techniques*, SIGGRAPH '87, pages 25–34. ACM.

Reynolds, C. W. (1999). Steering behaviors for autonomous characters.

Rosenzweig, M. L. and MacArthur, R. H. (1963). Graphical representation and stability conditions of predator-prey interactions. *The American Naturalist*, 97(895):209–223.

Stout, A., Konidaris, G. D., and Barto, A. G. (2005). Intrinsically motivated reinforcement learning: A promising framework for developmental robot learning. Technical report, Massachusetts Univ Amherst Dept of Computer Science.

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, Ł., and Polosukhin, I. (2017). Attention is all you need. In *Advances in neural information processing systems*, pages 5998–6008.

Vinyals, O., Babuschkin, I., Czarnecki, W. M., Mathieu, M., Dudzik, A., Chung, J., Choi, D. H., Powell, R., Ewalds, T., Georgiev, P., et al. (2019). Grandmaster level in starcraft ii using multi-agent reinforcement learning. *Nature*, 575(7782):350–354.

Vinyals, O., Fortunato, M., and Jaitly, N. (2015). Pointer networks. In *Advances in neural information processing systems*, pages 2692–2700.

Watkins, C. J. and Dayan, P. (1992). Q-Learning. *Machine learning*.

Yang, Y., Yu, L., Bai, Y., Wen, Y., Zhang, W., and Wang, J. (2018). A study of ai population dynamics with million-agent reinforcement learning. In *Proceedings of the 17th International Conference on Autonomous Agents and MultiAgent Systems (AAMAS)*, pages 2133–2135.