Alternative Learning Strategies for Collective Animal Movement

Toryn L.J. Schafer* Christopher K. Wikle†

Abstract

Collective behavior in natural animal systems is a frequent application for agent-based models. Realistic simulations of collective motion in traditional agent-based models is limited in complex environments and lacks notions of autonomy and memory. Reinforcement learning extends the definition of the agent to include perception and memory. The long term value of an agent's behavior is learned in reinforcement learning by interaction between the agent and its environment. We model this value using a double deep Q-network with a feed forward architecture. The reinforcement learning framework is demonstrated with a multi-agent environment with variable habitat types in a bounded environment. The agents learn to stay within the boundary and collectively move to a positive habitat.

Key Words: agent-based models, animal movement, collective behaivor, reinforcement learning, spatio-temporal, trajectory

1. Introduction

Schools of fish, flocks of birds, and herds of ungulates are a few examples of coordinated group behavior in natural animal systems. Coordinated collective movement processes are typically based on individual interactions and implemented with agent-based models. The simple individual-based rules employed in agent-based models lead to complex group dynamics. The rules attempt to reflect the internal processes of an agent, but generally imply automatic behavior and do not include memory (Ried et al., 2019). Specifically, the lack of autonomous learning by agents makes it difficult to incorporate interactions with habitat as every situation requires *a priori* assumptions about the agent's behavior in those specific environments. We propose to use a reinforcement learning framework to overcome the challenges of traditional collective movement models.

Typical agent-based models in collective animal movement modeling consider local rules. There are two general types of such rules: zonal and neighborhood based. Zonal rules place concentric circles or spheres around an agent in which a hierarchy of movement types are employed based on how many neighbors are within each zone (Couzin et al., 2002). Neighborhood rules simply tell an agent to adapt its movement to the average of a predefined number of nearest neighbors, as is commonly used in the self propelled particle model (Vicsek et al., 1995). Our reinforcement learning framework considers collective motion based on a hybrid of the zonal and neighbor-based rules where an agent perceives its neighborhood and the preferred movement is based on zonal distances (Morihiro et al., 2006).

Reinforcement learning is goal-oriented learning from the continuous interaction between an agent and its environment. The reinforcement learning framework extends the definition of an agent from typical agent-based models to include features of perception and memory (Ried et al., 2019). In reinforcement learning, perception by an agent is en-

*Department of Statistics, University of Missouri, Middlebush Hall, Columbia, MO 65211, USA. E-mail: tskb4@mail.missouri.edu

†Department of Statistics, University of Missouri, Middlebush Hall, Columbia, MO 65211, USA. E-mail: wiklec@missouri.edu

coded into the agent's state. Memory is incorporated by allowing the parameters controlling behavior to be learned from an agent's experience.

Lastly, the incorporation of the interaction between habitat and agents in a traditional collective movement model is not straightforward, but important for realistic inference (Strandburg-Peshkin et al., 2017). However, complex group dynamics in heterogeneous habitats can easily be incorporated in the reinforcement learning framework. We provide a straightforward implementation of reinforcement learning by multiple agents in a bounded space with variable habitat exhibiting collective behavior.

2. Reinforcement Learning

2.1 Markov Decision Process

The mathematical definition of reinforcement learning is based on the Markov decision process (MDP; Sutton and Barto, 1998). An MDP is defined as the tuple (S, A, T, γ, R) where:

- S: set of **states**
- $A = \{a_1, ..., a_k\}$: set of **actions**
- T: S x A x $S \rightarrow [0, 1]$ is a **transition probability function**
- $\gamma \in [0, 1)$ is a **discount factor**
- R: $S \rightarrow R$ is a **reward function**

For an MDP, a policy is a map for choosing an action in a given state, $\pi : S \rightarrow A$. The value function of a policy, π, for a state s, $V^{\pi}(s)$, is defined as

$$V^{\pi}(s) = E[\sum_{i=0}^{\infty} \gamma^i R(s_i) | s_0 = s], \tag{1}$$

which can be interpreted as the long term, discounted reward of following policy π from state s. Similarly, we can define a function based on the state-action pairs, $Q^{\pi}(s, a)$, as

$$Q^{\pi}(s, a) = E[\sum_{i=0}^{\infty} \gamma^i R(s_i) | s_0 = s, a_0 = a], \tag{2}$$

which can be interpreted as the long term, discounted reward of following policy π after taking action a in state s.

The goal of reinforcement learning is to learn the policy, π, for which the value function is maximized. A policy $\pi = \pi^*$ is said to be optimal if the value function, and hence the Q-function, is maximized $\forall\ s \in S$, i.e.

$$\pi^*(s) \in arg\max_{a \in A} Q^{\pi}(s, a). \tag{3}$$

The above expression is referred to as Bellman optimality. The resulting expression for the Q-function under Bellman optimality is

$$Q^{\pi^*}(s, a) = R(s) + \gamma \max_{a' \in A} Q^{\pi^*}(s', a'), \tag{4}$$

where s' is the next state. The above is referred to as the Bellman optimality equation (Sutton and Barto, 1998).

3. Model and Method

The reinforcement learning problem is solved by iteratively estimating the Q-function from an agent's interactions with the environment. Let s_t and a_t denote the state-action pair for an agent at iteration t. In problems with relatively small state-action spaces, the most common algorithm is Q-learning. Q-learning updates the current estimate of the Q-function of a state-action pair by the following

$$Q(s_t, a_t) \leftarrow Q(s_t, a_t) + \alpha[R(s_{t+1}) + \gamma \max_{a' \in A} Q(s_{t+1}, a') - Q(s_t, a_t)], \tag{5}$$

where α is a learning rate (Sutton and Barto, 1998).

In problems with larger state-action spaces, the updating of the Q-function in 5 is typically replaced with a parameterized model which learns the relationship between state-action pairs and the Q-function. The prediction of the model is then used to choose the next action. To illustrate using the notation of Van Hasselt et al. (2016), let the Q-function be a function of the state-action pair and a set of unknown parameters, $\boldsymbol{\theta}$, then

$$Y_t^Q \equiv R(s_{t+1}) + \gamma \max_{a' \in A} Q(s_{t+1}, a'; \boldsymbol{\theta}_t) \tag{6}$$

is the target value learned by the model where $\boldsymbol{\theta}_t$ is the estimate of the parameters at iteration t (Van Hasselt et al., 2016).

3.1 Double Deep Q-Network

A deep Q-network (DQN) algorithm uses a deep neural network model to estimate the relationship between state-action pairs (input) and the Q-function (Equation 6) with two novel features: experience replay and the use of a target network (Mnih et al., 2015). Experience replay refers to storing observed transitions, (s_t, a_t, r_t, s_{t+1}), and randomly sampling the observed transitions for updating the online network. In 6, the target, Y_t^Q, is calculated using the online values of parameters, $\boldsymbol{\theta}_t$. The target of the DQN is calculated from a copy of the online network called the target network with parameters $\boldsymbol{\theta}^-$ where the parameters of the target network are copied from the online network only every τ steps:

$$Y_t^{DQN} \equiv R(s_{t+1}) + \gamma \max_{a' \in A} Q(s_{t+1}, a'; \boldsymbol{\theta}^-) \tag{7}$$

The DQN uses the same estimated model for action choice and policy evaluation which makes it more prone to overfitting and overoptimistically estimating the value. Van Hasselt et al. (2016) proposed the double DQN algorithm to improve the accuracy of the value estimation. The target of the double DQN,

$$Y_t^{doubleDQN} \equiv R(s_{t+1}) + \gamma Q(s_{t+1}, \operatorname*{argmax}_{a' \in A} Q(s_{t+1}, a'; \boldsymbol{\theta}_t); \boldsymbol{\theta}'_t) \tag{8}$$

uses online network with parameters $\boldsymbol{\theta}_t$ to select the action and a second network with parameters $\boldsymbol{\theta}'_t$ to evaluate the value.

3.2 Agent Environment

For our simulation, we incorporated three components into the environment: a boundary, positive and negative habitat zones, and N agents (Figure 1). We assumed the agent could perceive other agents, the boundary, the current habitat zone, and its coordinate position. Therefore the state, $\mathbf{s} = (s_1, s_2, s_3, s_4, s_5)'$, for each agent was a 5 dimensional vector where:

- $s_1 = d \in \mathbb{R}$ is the distance to a neighbor perceived with probability $d^{-\beta}$ (Morihiro et al., 2006),
- $s_2 = b \in \mathbb{R}$ is the distance to the nearest boundary,
- $s_3 = c \in \{0, 1, 2, 3\}$ where the indices correspond to outside the boundary, neutral zone, negative zone, and positive zone, respectively, and
- $(s_4, s_5) = (x, y) \in \mathbb{R}^2$ is the position of the agent.

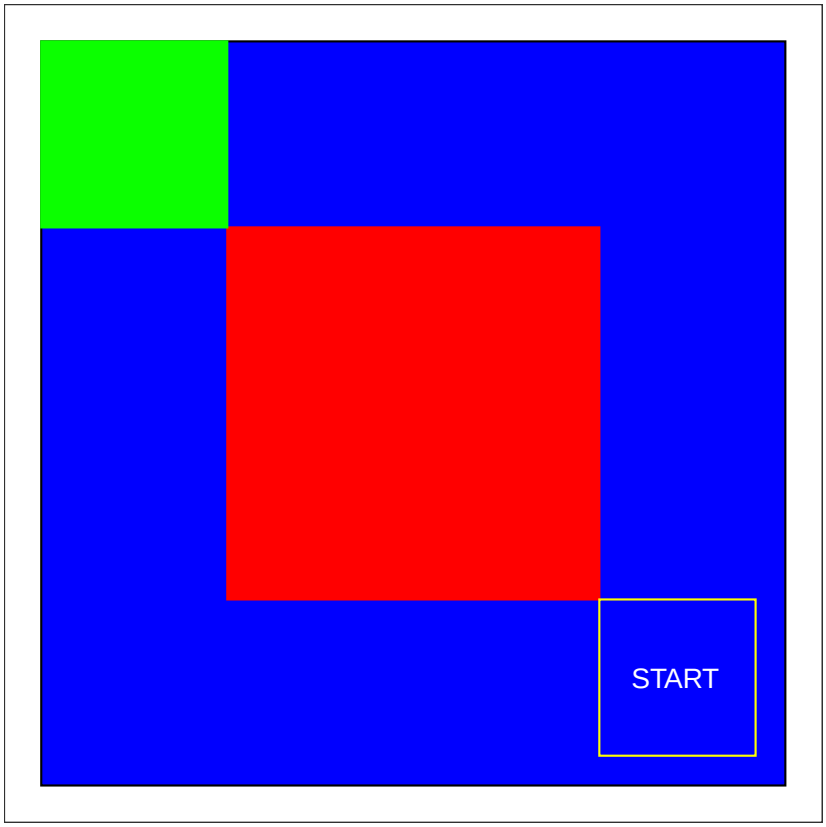

Figure 1: This graphic depicts the environment. The blue area is the neutral zone. The red area is the negative zone. The green area is the positive zone. All agents are initiated to start at a random point within the yellow square.

The actions space consisted of eight unit directional vectors where the first four correspond to collective motion and the last four correspond to staying within the boundary. The assumption is that the agent is moving one body length at each step represented by the unit directional vector (Morihiro et al., 2006).

- a_1: attraction to neighbor,
- a_2: positive parallel alignment with neighbor,
- a_3: negative parallel alignment with neighbor,
- a_4: repulsion to neighbor,
- a_5 : attraction to nearest boundary,

- a_6 : parallel alignment to nearest boundary toward positive habitat zone,
- a_7 : parallel alignment to nearest boundary away from positive habitat zone, and
- a_8 : repulsion to nearest boundary.

The boundary alignment actions were chosen such that the agent could choose the same action, a_6, to go toward the positive zone regardless of which side of the environment it is located. This makes learning easier than using just clockwise and counterclockwise directions for boundary alignment.

The direction between the current and next position of the agent was then calculated as a weighted average of the agent's current direction and the chosen action unit directional vector given inertia

$$\boldsymbol{m}_{it} = \frac{(1-\kappa)\boldsymbol{m}_{i,(t-1)} + \kappa \boldsymbol{m}_a}{|(1-\kappa)\boldsymbol{m}_{i,(t-1)} + \kappa \boldsymbol{m}_a|} \tag{9}$$

where $\boldsymbol{m}_{it}$ is the direction vector for agent i at time t, $\boldsymbol{m}_a$ is the directional vector for the action a, and κ is a parameter which controls inertia (Morihiro et al., 2006).

3.2.1 Reward System

The immediate reward received by the agent after taking an action is the mean of three values. The three values are associated with the collective behavior, tank boundary, and habitat zone. The magnitude and direction of the rewards are the biggest influence on the final learned policy. The reward system for the collective behavior is a zonal type system similar to that used by Couzin et al. (2002) with a zone of repulsion, alignment, and attraction based (Table 1). Outside of these three zones, an agent cannot perceive a neighbor. Again, since the actions are normalized, all the distances used in the reward system are interpreted as distances in body lengths. The values used to delineate the different zones are the same as used by Morihiro et al. (2006).

The agent is rewarded for staying within the tank boundary. The agent can only perceive or not perceive the boundary based on the distance to the nearest edge (Table 2). If the agent perceives the tank boundary, it is penalized for moving toward the boundary (action a_5) and rewarded for taking actions a_6, a_7, or a_8. This reward system allows agents to learn to travel near the boundary, but not cross over it.

Finally, the rewards for the different habitat zones are based on the zone observed after the agent executes the chosen action. The rewards for the different zones are -10, 0, -1, and $+5$ for outside the boundary, neutral zone, negative zone, and positive zone, respectively.

4. Simulation

We simulated 10 agents in the environment. An episode was defined to be the minimum of the number of steps observed before any agent moved outside the boundary and 200 steps. The MDP discount parameter, γ, was set to 0.8.

The state and action sequences were evaluated asynchronously for each agent. For calculating the nearest neighbor distance in the state vector, the perception parameter β was set to 0.5. In order to encourage exploration of the action space early in learning, we used an ϵ-greedy action selection with decaying exploration parameter ϵ. Under an ϵ-greedy action policy, the agent chooses a random action with probability ϵ or the best action based on the current Q-function with probability $1-\epsilon$. The probability ϵ decays to zero

Table 1: Reward system for actions related to collective behavior by zones defined by neighbor distance. The immediate reward received for taking one of the collective behavior actions is based completely on the distance to perceived neighbor in the current state, d.

		Action			
Zone	**Neighbor Distance**, d	a_1	a_2	a_3	a_4
Repulsion	$d < 4$	-1	-1	-1	+1
Alignment	$4 < d < 20$	-1	+1	-1	-1
Attraction	$20 < d < 50$	+1	-1	-1	-1
No Perception	$d > 50$	0	0	0	0

Table 2: Reward system for actions related to tank boundary by distance to nearest tank wall. The immediate reward received for taking one of the tank boundary actions is based completely on the distance to nearest boundary in the current state, b.

	Action			
Boundary distance, b	a_5	a_6	a_7	a_8
$b < 4$	-1	+1	+1	+1
$4 < b$	0	0	0	0

each episode. Since the agents were defined identically, each state-action sequence for the 10 agents was treated as an independent replicate from the environment and stored in the replay memory.

The deep neural network model used for the double DQN was a feed forward model with 3 hidden layers of size 64, 32, and 16 nodes, respectively. The parameters of the online neural network were updated each episode with a minibatch of 128 samples randomly selected from the replay memory. The parameters of the evaluation model in the double DQN were copied from the action selection model every 20 episodes and the simulation was run for 500 episodes. It is worth noting that the differences in using a DQN and double DQN were negligible for the proposed agent environment. This suggests learning the optimal policy in the scenario was not being severely overfit by a DQN.

5. Results

The agents quickly learn to stay within the boundary for the set maximum of 200 steps per episode after about 70 episodes and consistently have episode lengths of 200 after about 280 episodes 2.

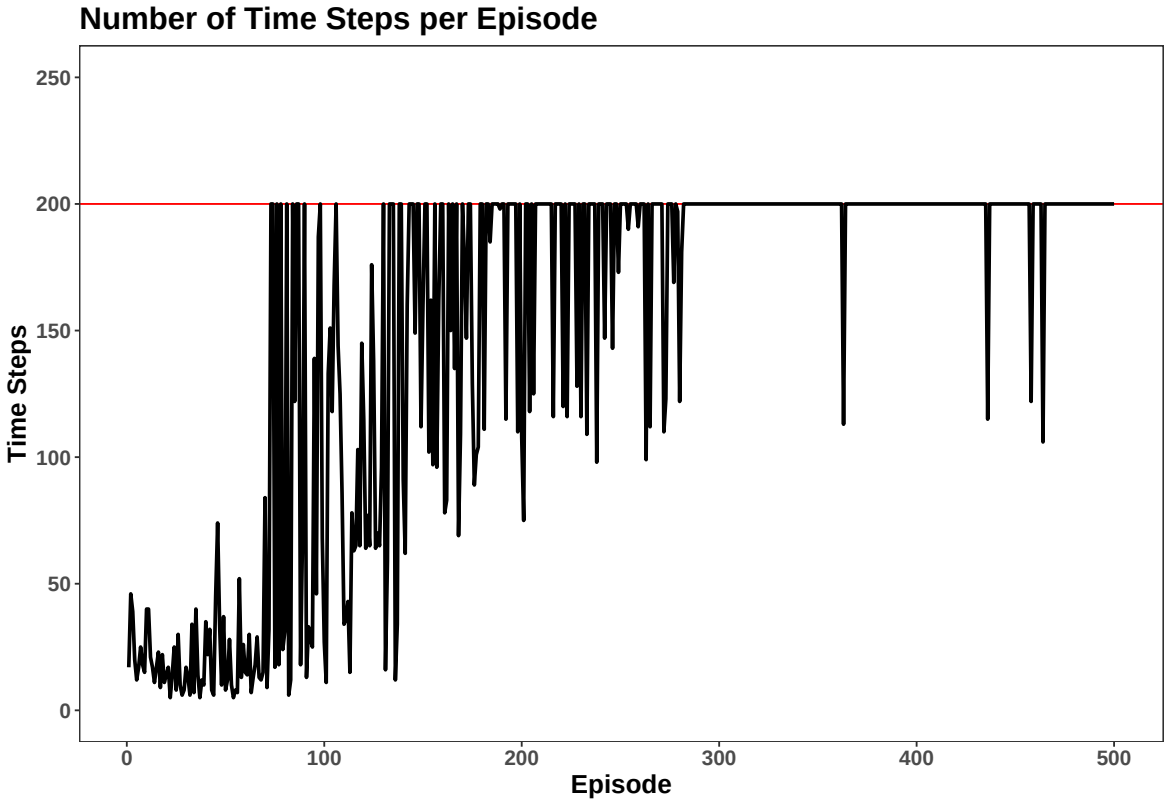

Figure 2: The episode length is defined as the minimum of the number of steps before an agent passes the boundary or 200. The agents quickly learn to stay within the boundary given the frequency of episodes with 200 steps after about episode 70.

The optimal policy learned by the agents encouraged collective movement as one group toward the positive habitat zone. Once the agents were in the positive habitat zone, the collective movement was secondary to staying in the positive habitat zone and therefore smaller groups were able to form.

Figure 3 shows the progression of learning by the agents. At episode 70, the movements appear to mostly be random with some small groups forming. At episode 250, the agents have learned to stay in the boundary for the period of 200 steps, but do not end up in the positive habitat zone. Under the final policy, the agents travel as a group to the positive habitat zone and smaller groups form as they trade off the reward for the positive zone and collective behavior. We provide animations of the sampled behavior for episode 70, episode 250, and the final policy at the following github page: https://github.com/schafert/JSM2019Proceedings.

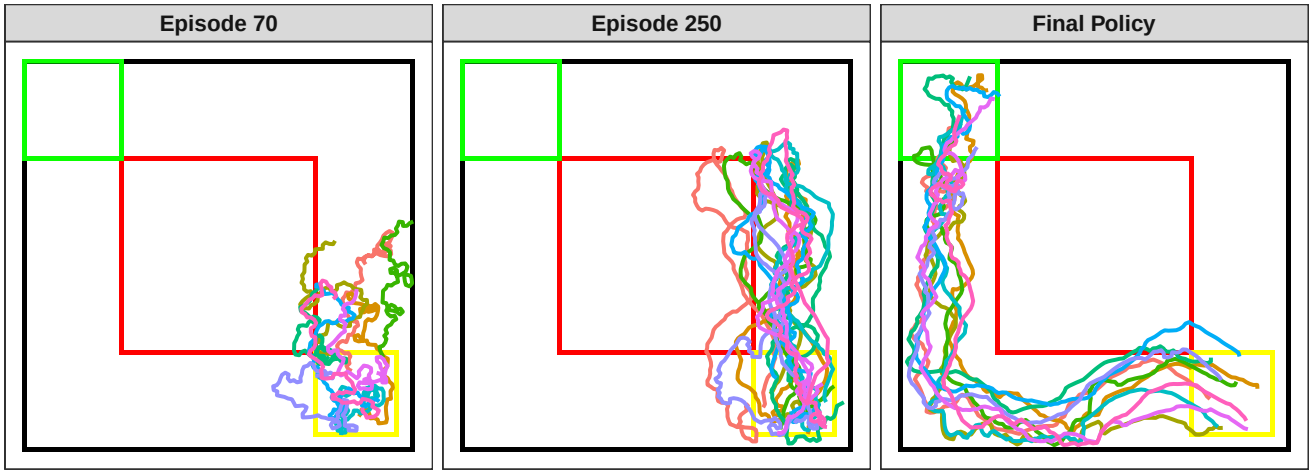

Figure 3: The behavior trajectories of the 10 agents in episode 70, episode 250, and under the final policy. The colored boxes delineate the different habitat zones as shown in Figure 1

6. Conclusions

Reinforcement learning allows for simulated collective movement between agents in a complex environment. In our simulation, reinforcement learning agents learned the optimal long term behavior in a variable environment rather than *a priori* establishing agent based rules as in traditional agent-based models. We used deep models for learning the long term discounted value of behavior choices.

There are many possible extensions to the current work. First, the deep model used in the double DQN was simply a feed forward neural network. The sequential nature of the agent trajectories suggest that it might be more plausible to use a recurrent neural network. Second, the degree of collective behavior exhibited in real world systems are likely to be a function of more than the current habitat. For example, there are seasonal trends in animal grouping behavior for migratory species. The different behavior types could be incorporated into the reward function for collective behavior. Lastly, the reinforcement learning framework could shed light on the effects of habitat fragmentation by evaluating agent behavior in novel environments.

References

Couzin, I. D., Krause, J., James, R., Ruxton, G. D., and Franks, N. R. (2002). Collective memory and spatial sorting in animal groups. *Journal of theoretical biology*, 218(1):1–11.

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.

Morihiro, K., Isokawa, T., Nishimura, H., and Matsui, N. (2006). Emergence of flocking behavior based on reinforcement learning. In *International Conference on Knowledge-Based and Intelligent Information and Engineering Systems*, pages 699–706. Springer.

Ried, K., Müller, T., and Briegel, H. J. (2019). Modelling collective motion based on the principle of agency: General framework and the case of marching locusts. *PloS one*, 14(2):e0212044.

Strandburg-Peshkin, A., Farine, D. R., Crofoot, M. C., and Couzin, I. D. (2017). Habitat and social factors shape individual decisions and emergent group structure during baboon collective movement. *Elife*, 6:e19505.

Sutton, R. S. and Barto, A. G. (1998). *Introduction to reinforcement learning*, volume 2. MIT press Cambridge.

Van Hasselt, H., Guez, A., and Silver, D. (2016). Deep reinforcement learning with double q-learning. In *Thirtieth AAAI conference on artificial intelligence*.

Vicsek, T., Czirók, A., Ben-Jacob, E., Cohen, I., and Shochet, O. (1995). Novel type of phase transition in a system of self-driven particles. *Physical review letters*, 75(6):1226.