Mimicking the Bat Brain

Reinforcement Learning Project (2020 - 2021)

Dorian Desblancs
École Normale Supérieure Paris-Saclay
Master MVA (Mathématiques, Vision et Apprentissage)
4 Avenue des Sciences, 91190 Gif-sur-Yvette, France
dorian.desblancs@mail.mcgill.ca

Abstract

This project explores how echolocation can be incorporated in an agent's learning process for spatial navigation. We use the Foster model [1] as the basis of our learning algorithm. This model uses temporal difference learning and an actor-critic model to learn the optimal way to get to a specified platform. It has been proven to behave similarly to rats on path navigation tasks. We inspire our echolocation system by that used by bats for object recognition and detection, and alter the Foster algorithm in order to process echolocating sound waves at each step. Our results demonstrate that the learning process can be greatly sped up by incorporating echolocation in the learning process. This study further highlights the need for more interactive agents in reinforcement learning, especially in fields such as autonomous path navigation that aim to be applied in the real world one day.

1. Introduction

Motion planning, or path planning, is a field of research that aims to reduce travel time or path length between two points. It is a very old field, that has inspired countless classic algorithms such as Dijkstra's algorithm and rapidly exploring trees. In recent years, the field has grown exponentially due to its applications in robotics and automation. Most notably, determining the best path to get to an unknown location has proven to be a very hard problem.

Autonomous path navigation has also been a popular idea in the field of neuroscience, both computational and cognitive. Over the years, researchers have attempted to uncover the mysteries behind how various mammalian species move throughout the world [2]. The reasons for this are two-fold. First, knowing how mammals navigate has allowed us to better understand the brain. This task involves a variety of areas of the cerebrum, and can be motivated by countless reasons, such as foraging for food or going to see another individual. Second, our understanding of mammalian navigation has led to a variety of popular path navigation algorithms, that try to mimic real behavior. These algorithms have offered us great insights into how brains process information. More broadly, the science behind how species learn new information is at the heart of fields such as meta-learning and reinforcement learning.

In this project, we explore a classic path planning model: the Foster path navigation model [1]. This model proposes a new algorithm for path navigation that relies upon temporal difference learning, or TD-learning. The model incorporates an actor-critic algorithm in order to update the different parameters. The Foster model was introduced in order to mimic a rat's performance on the watermaze task. This task is extremely popular in neuroscience, and was devised in order to study the psychological processes and neural mechanisms of spatial learning and memory [5]. In it, an animal would be inserted into a large water basin that contains a platform. The speed at which the animal reaches the platform would then be recorded. This task would be recorded time and time again on a variety of experimental setups to determine animal navigation behaviour. The Foster model aimed to mimic rat performance on the watermaze task.

In this project, we first reproduced the results obtained by the original paper. We then proposed a simple alteration to the algorithm that aims to incorporate bat echolocation throughout the learning process. Our results show that the incorporation of echolocation in the learning process greatly decreases the time it takes a learning agent to discover the optimal path to a goal. This study highlights the need for more interactive reinforcement learning, in which an agent can learn from its environment using more than just state/action policies. This is especially the case for tasks such as path planning, that aim to be used reliably in the real world. The code for our various experiments can be found here [1].

[1] https://github.com/d-dawg78/MVA_RL

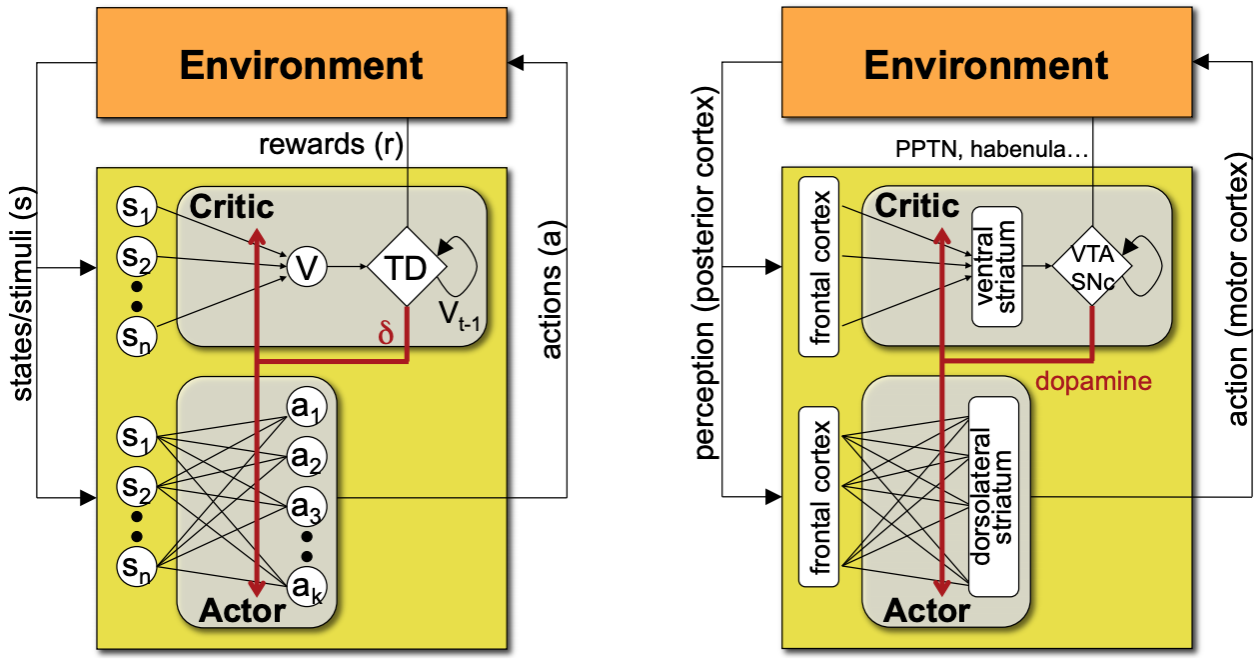

Figure 1. The Actor-Critc Architecture in both Reinforcement Learning and the Brain [6]

2. Background Information

2.1. Reinforcement Learning

2.1.1 Temporal Difference Learning

Historically, reinforcement learning algorithms were created to learn how to perform complex behaviour on their own. These algorithms only used reward and punishment errors as their teaching signals [8].

The TD-learning algorithm was published by Richard Sutton in 1988. The idea behind the algorithm is simple: at each step, the TD algorithm takes an action and predicts its reward at the next moment in time. When the next moment comes, the predicted and actual reward are compared. If these are different, the algorithm adjusts its old predictions to be closer to the new ones. Note that the rewards must be given in advance for the algorithm to work.

More formally, the TD error is given by the following equation:

$$\delta_t = r_{t+1} + \gamma V(s_{t+1}) - V(s_t) \tag{1}$$

where:

- V denotes the chosen value function
- s_t denotes the state at time t
- r_t denotes the reward at time t.

Over time, the algorithm's predicted rewards at each step become more accurate. Its policy is optimized accordingly.

2.1.2 Actor-Critic Methods

Actor-Critic methods are derived from TD learning. The difference: actor-critic models represent the policy and value function independently [8]. Figure 1 displays the method's architecture.

The actor represents the policy structure. It is used to select the actions that the algorithm should take at each step in order to reach a desired goal. The critic, on the other hand, estimates the value function. Its role is to constantly criticize the policy taken by the actor. It usually takes the form of the TD error, displayed in the previous section. At each step, when this error is positive, the critic suggests that the action be strengthened, so as to be chosen more frequently. The opposite happens when the error is negative.

2.2. Neuroscience

In the mid-1990s, Montague et al. proposed one of the most groundbreaking theories in neuroscience. They noticed that the firing patterns some dopamine neurons displayed were correlated with the rewards animals were trained to expect [4]. When animals were trained to receive a reward using a certain cue, dopamine neurons would fire at a higher rate at the beginning of training. These would

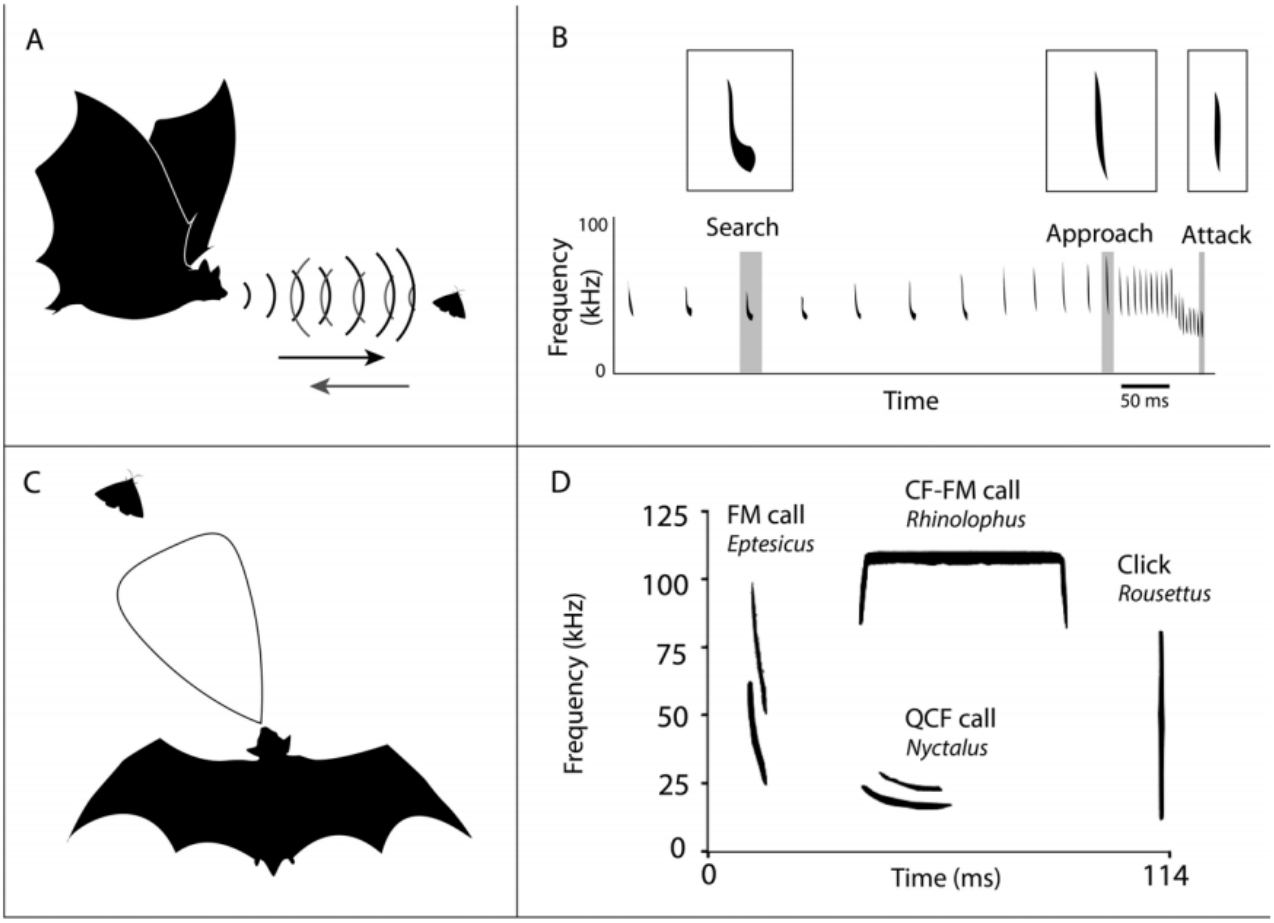

Figure 2. Principles of Echolocation. (A) Bats emit sound signals and analyse the returning echoes to perceive their environment. (B) Spectrogram – frequency over time representation, showing a sequence of signals emitted by a bat that is approaching a prey item (C) Bats can steer their echolocation beam in space and thus control their sensory acoustic gaze. (D) Three very different types of signals, representing most echolocation signals used by bats. Figure from [9]

.

fire less frequently as the animal grew accustomed to the reward. On the other hand, when the reward was suddenly removed, the dopamine neurons would go back to firing.

Montague et al. [4] suggested that dopamine neurons encode a temporal difference reward prediction error. This theory was proven to be correct in the following years [7]. The Foster model for spatial navigation [1] inspires itself from these studies, and supports the claim that a form of TD learning is indeed implemented throughout the brain. We will present its intricacies later in this report.

2.3. Bat Echolocation

Bats are arguably the most interesting species on Earth. They live in caves, look horrific, and host a vast array of viruses. Yet, their spatial navigation system is one of the most complex and refined mechanisms that can be found in nature. Bats use echolocation in order to localize, detect, and recognize objects that surround them [3]. Every second, they emit cries in the range of 14 000 to 100 000 Hz, using either their mouth or nose depending on the species. A single cry can last between 1 and 200 ms. Let us now explore the mechanism in more detail.

The second figure above displays the frequency of a bat cry as it approaches an object (prey in this case). *(Q)CF* denotes (Quasi) Constant Frequency while *FM* denotes Frequency Modulation. Notice how the cries' frequency ranges becomes larger as the bat approaches an object? Notice how the cries' durations are shortened as the bat approaches the object? This is the essence of bat navigation.

As the bat flies forward, it emits sound waves. These

are then reflected. However, the sound's frequency is altered due to the Doppler effect. More precisely, suppose a bat flies with velocity v_{bat} and emits a sound at frequency w_{emit}, then the frequency of the echoed sound is:

$$w_{observed} = w_{emit} \frac{v_{sound} + v_{bat}}{v_{sound} - v_{bat}}. \quad (2)$$

Hence, when the bat is flying forward, the echoed frequency it perceives is higher than the one it emitted. The delay between sound emission and reception indicates the distance between them and the object the sound was reflected off. Even better, the pattern the reflected sound waves follow is indicative of the type of object the bat is surrounded by. Say the object in front of it were a rock. The sound waves would be reflected off the rock in constant fashion, with regular frequency shifts. On the other hand, say the object were a moth. Its wing movements would reflect the bat's sound waves very differently at each time step. The reflected waves would however follow a pattern through time, allowing the bat to recognize the object as a moth!

Now, as a bat approaches an object, it must still be able detect the object's distance. It therefore modulates the outgoing sound's frequency and shortens the sound's duration. This allows the bat to continue to recognize the object. Say the sound were long and at a constant frequency when the object is very close. In this case, the bat would be emitting a sound while perceiving a reflected signal. This would be highly problematic. By emitting short, frequency-modulated signals, they can detect the object's location and properties very precisely. For more information, we encourage the reader to consult [9].

3. Methodology

3.1. Environment

The watermaze environment we used is illustrated in Figure 3. It has a radius of 60 centimeters. The platform our agent tries to reach has a radius of 10 centimeters. The objects can be generated using a given size. In our case, we limited the objects to 4 squares of width 25, 15, 10, and 8 centimeters. When the agent tries to move outside the boundary or inside an object, it is reflected off it. In each of our trials, we limited the length of an episode to 60 moves. We found this number to offer enough time for exploration at each episode. At each step, we allowed the agent to move 5 centimeters in a desired direction. We allowed either 8 or 16 equally-spaced total directions.

The sound waves are generated at each step, and are allowed to move 60 times. They are reflected in the same fashion as the agent. If they hit the agent, however, they stop moving throughout the environment. Note that we did not attempt to reproduce the acoustic properties of real waves (we did not take into account sound absorption, for instance). Bats are able to detect very weak sounds in nature due to their ear sensitivity. More importantly, this project was dedicated to generating results that capture echolocation's potential in reinforcement learning. The acoustic correctness of the environment is left for future work.

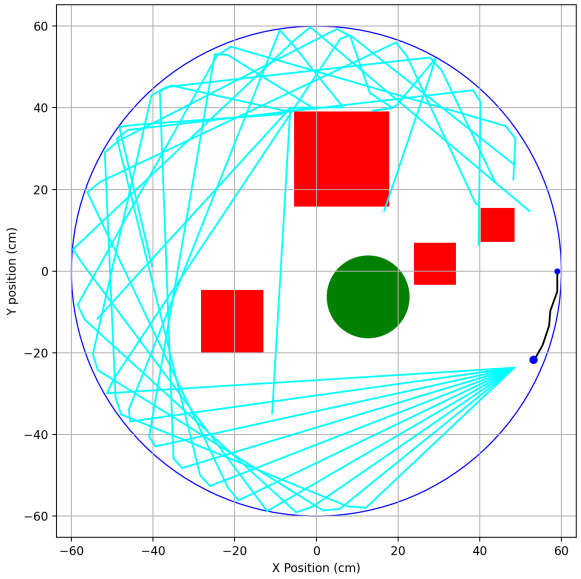

Figure 3. Sample Environment. (Cyan) Sound waves generated by agent. (Red) Objects. (Green) Platform to reach. (Blue) Watermaze limits.

.

3.2. The Foster Model

This section explores the elements that make up the Foster spatial navigation algorithm.

3.2.1 Hippocampal Place Cells

In neuroscience, place cells are described as neurons mapped to a particular location in space. They are activated at each time step, allowing the agent to infer its location and potential next move.

In this project, we map each place to a location in the environment. We call this location the place cell's center. Each place cell then has a standard deviation value of 3 centimeters. We used a total of $N = 493$ place cells. These values were found to work well, and are fairly similar to the original paper [1]. In the Foster model, place cells activations are modelled as Gaussian functions of locations in the maze. When the agent is at position p, each place $i \in 1, ..., N$ is activated using:

$$f_i(p) = exp\left(-\frac{||p - s_i||^2}{2\sigma^2}\right) \quad (3)$$

where s_i is place cell i's center and $\sigma = 3$ as noted

above. During each time step, place cells are activated once a new location has been determined according to the algorithm's policy.

3.2.2 The Critic

Once a reward has been determined for the agent's new location, the critic is updated. As noted previously, the critic follows the TD error equation. We denote the critic as w. It is a vector of length N. After place cells have been activated, the critic computes a weighted sum of the place cell inputs $f_i(p)$. This is given by:

$$C(p) = \sum_i w_i f_i(p) \tag{4}$$

This weighted sum is computed for both the actual and predicted location place cell activations. The key to the Foster model lies in the fact that $C(p)$ is used as the value function for this actor-critic model. The prediction error is thus:

$$\delta_t = R_{t+1} + \gamma C(p_{t+1}) - C(p_t) \tag{5}$$

where R_{t+1} is the reward reward obtained by the predicted location p_{t+1}. As the model is trained, we converge towards:

$$C(p_t) = \bar{R}_{t+1} + \gamma C(p_{t+1}) \tag{6}$$

Note that the critic is updated using:

$$w_i = w_i + \eta \delta_t f_i(p_t) \tag{7}$$

In the project, the learning rate η was set to 0.5 for fast learning. The discounting factor γ was set to 0.99.

3.2.3 The Actor

At the beginning of each episode, the actor generates a probability distribution that encodes the probability of choosing each action. In our case, this corresponds to 8 or 16 directions to move in. This probability distribution is computed using the place cell activations of the current location.

Assuming the actor is encoded by a matrix z of dimension $8 \times N$ or $16 \times N$ (depending on the number of movement directions), the actor first outputs an action vector using:

$$a_j(p) = \sum_i z_{ji} f_i(p) \tag{8}$$

where $j = 1, ..., 8$ or 16.

This vector outlines the agent's relative preference of using direction j at location p. The probability distribution encoding preferences for each action is then computed using:

$$P_j = \frac{exp(2a_j)}{\sum_k exp(2a_k)} \tag{9}$$

Note that we used the exp-normalize trick in order to avoid overflow when computing P. The predicted move at each time step is calculated using the distribution P.

Once the TD error has been computed, the matrix z is updated according to:

$$z_{ji} = z_{ji} + \eta \delta_t f_i(p_t) \tag{10}$$

Note that only row j is updated. j corresponds to the chosen move.

3.2.4 The Coordinate System

In their paper, Foster et al. also introduce a coordinate system to supplement the actor-critic model. This coordinate system is introduced in order to help the agent quickly adapt to changes in the environment, most notably changes in the platform location after a certain number of episodes.

The coordinate system essentially consists of two networks. The first learns X coordinates. The second learns Y coordinates. Both use the following equations to learn these coordinates given an array of place cell activations (computed for predicted position p at each step):

- $X(p) = \sum_i w_i^X f_i(p)$
- $Y(p) = \sum_i w_i^Y f_i(p)$

Once the agent has reached the platform, the coordinate system sets X' and Y' to the platform coordinates. These are the goal coordinate memory. The coordinate system can then be seen as another direction that can be chosen at each time-step. Whenever it is chosen, it uses $X(p)$ and $Y(p)$ and its goal coordinate memory to choose the closest direction of movement. This can be done in multiple ways. In our code, we used the directions d_x and d_y computed such that $d_x = X' - X(p)$ and $d_y = Y' - Y(p)$. We then use the cosine similarity between d_x and d_y to determine the direction of movement.

Once the reward and critic have been updated, $X(p)$ and $Y(p)$ are computed for the next time step using the TD algorithm's predicted position. Finally, the coordinate weights w_i^X and w_i^Y are updated using:

$$w_i^X = w_i^X + \eta(\Delta x_t - (X(p_{t+1}) - X(p_t))) \sum_{k=1}^{t} \lambda^{t-k} f_i(p_k) \tag{11}$$

$$w_i^Y = w_i^Y + \eta(\Delta y_t - (Y(p_{t+1}) - Y(p_t))) \sum_{k=1}^{t} \lambda^{t-k} f_i(p_k) \tag{12}$$

where Δx_t and Δy_t denote the difference between current and new position at each time step.

The coordinate dimension in the actor's weight matrix is then only updated when the coordinate move is chosen. This is done by adding the TD error multiplied by the learning rate to each of its N values.

Do note that the coordinate system is quite heavy computationally. Most notably, updating its weights involves summing each one of the place cell's past activations. However, once the coordinate is tuned, its moves quickly become optimal. Hence, as the number of training episodes grows, the probability of choosing the coordinate system is increased and is very close to 1. For more details about the motivations behind this system, and its neurological foundations, please consult the original Foster paper [1].

3.3. Echolocation

In this project, we were only able to approximate echolocation. This was done in the following way.

At each time step, 10 sound waves were generated. These were generated in the direction the agent is currently facing. Note that we limited the number of sound waves to 10 due to computational constraints. Our code allows the user to use as many sound waves as one wants. We then rebalanced the action probabilities at each time step according to the sound waves that are reflected upon the agent. If the sound wave is reflected off a wall or an obstacle, the probability of moving to the direction nearest to the wave's incoming direction is set to 0. If, however, the sound wave is reflected off the platform, the probability of moving to the direction nearest to the wave's incoming direction is left untouched and all other action direction probabilities are set to 0. Note that the probability distribution is re-set such that it sums to 1 whenever re-balancing occurs. Also note that the coordinate system's probabilities are always left untouched. In some sense, the coordinate system acts as a memory coordinate that takes over once enough episodes have taken place.

This slight alteration to the Foster algorithm mimics how a bat navigates. Its auditory system is capable of distinguishing objects according to the reflected sound wave's properties. In this project, we skip the *learning process* step (i.e. the step during which a bat learns how to distinguish objects by their sound wave patterns or properties through trial and error). We assume that our bat is fully grown, knows how to recognize obstacles, walls, and platforms, and navigates according to the TD algorithm. Our results are as follows.

4. Results

In order to test each algorithm, we separated training episodes into days. Each day corresponds to 100 episodes. The mean and standard deviation path length for each day's episodes was then computed, and plotted for this report. Note that we also plotted the mean and standard deviation collision count when comparing our echolocation algorithm's performance to the regular Foster model. These plots can be found in the Appendix.

We also changed the platform and obstacles' locations every 8 days. Our goal was to measure how fast each algorithm was able to adapt to a new environment. We also wanted to know to what extent incorporating echolocation in the Foster model would help our agents adapt to a changing environment. Note that the escape latency denotes the number of seconds a trial lasts. The maximum escape latency here is 59 (this means the agent did not manage to reach the platform during the trial).

4.1. The Foster Model

As one can see in Figures 4 and 5, adding the coordinate system to the TD learning algorithm helps with performance. The coordinate system helps the algorithm converge towards an optimal policy faster, especially when the platform's location is suddenly changed. We did not, however, observe the performance gain Foster et al. [1] achieved.

This is most likely due to the fact that we bounded the length of an episode to 60 seconds. In contrast, they only ended their trials when the agent had reached the platform. This means that, especially in the first few episodes, the vanilla TD algorithm performed much worse than the coordinate system algorithm. The main advantage of the coordinate system lies in the first few iterations, when a platform has changed locations. Learning a goal coordinate allows the algorithm to adapt to change much more quickly. This can most notably be seen in the last 8 days of the obtacle watermaze (Figure 5). The vanilla TD algorithm is stuck at a mean escape latency of more than 50 seconds. In contrast, the coordinate system algorithm has already reached a mean escape latency of less than 30 seconds.

4.2. Echolocation

In contrast, we found a drastic improvement in both the vanilla TD and Coordinate System algorithms when incorporating our echolocation model. When looking at Figures 6, 7, 8, and 9, one can notice that the model that uses echolocation in conjunction with its learning algorithm substantially outperforms the regular learning algorithm. This is the case for both the obtacle and regular watermaze.

These results are normal. By incorporating echolocation in the Foster model, we are able to restrict the action space. This restriction of the action space mostly allows the algorithm to escape performing useless actions at each step. More rarely, the optimal action is selected when the platform reflects one of the agent's sound waves. In this case, the agent almost always travels straight to the platform due to ensuing time steps' sound waves pointing directly at the

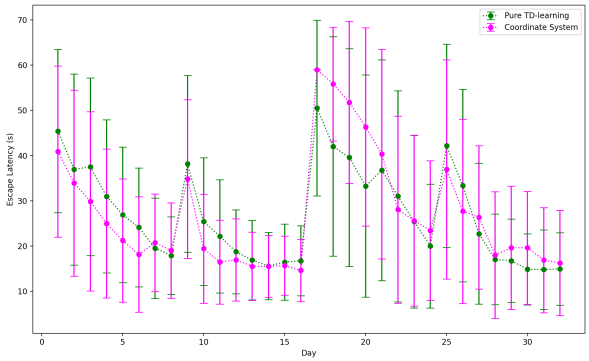

Figure 4. Pure Foster Model Performance Comparison on the Regular Watermaze

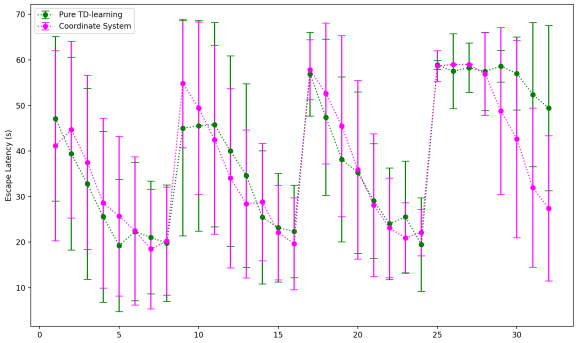

Figure 5. Pure Foster Model Performance Comparison on the Obstacle Watermaze

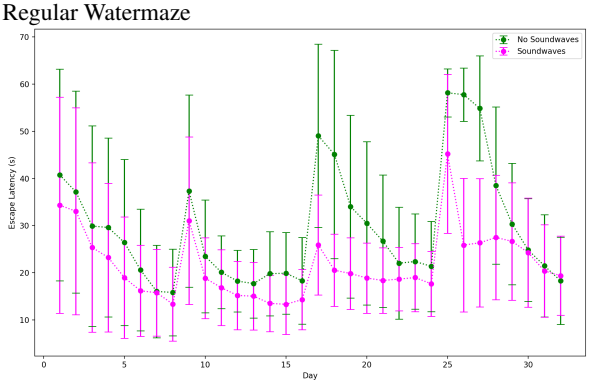

Figure 6. TD Learning Performance Comparison on the Regular Watermaze

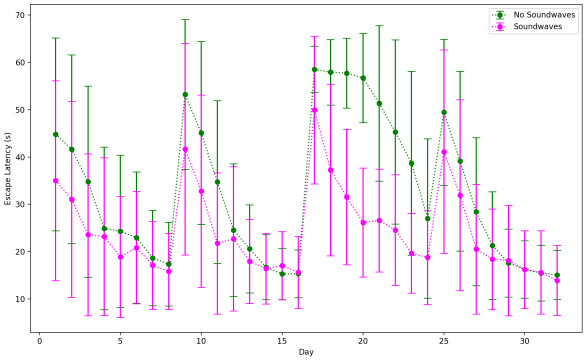

Figure 7. TD Learning Performance Comparison on the Obstacle Watermaze

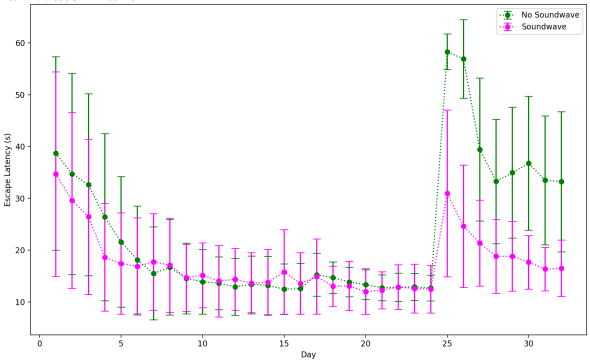

Figure 8. Coordinate System Performance Comparison on the Regular Watermaze

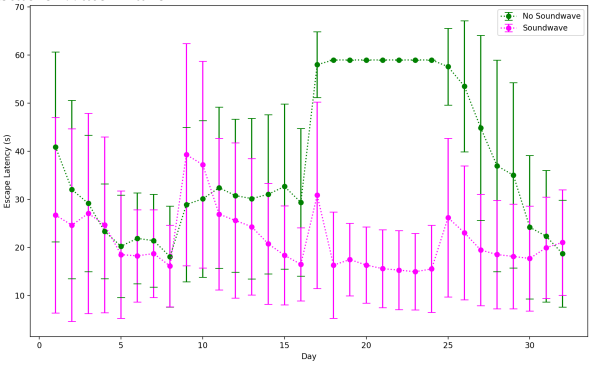

Figure 9. Coordinate System Performance Comparison on the Obstacle Watermaze

goal state.

Do note that the results presented here are quite limited. These highlight how much faster an agent learns when it is able to interact with an environment using echolocation. We hope to report more in depth results in the future. Most notably, counting the average number of times probability re-balancing occurs during an episode would allow us to infer how many moves are eliminated from consideration at each step. Using a simple eye test, this number seems to be around 2. These numbers do however deserve to be known so that we can judge how helpful echolocation is in our experiments.

5. Future Work

The main improvement we see in our approach to echolocation for path navigation is the following.

In this project, we assume that our agent already knows how to distinguish objects from each other. The reality of most learning agents and species on this Earth is however quite different. We learn through trial and error. At the same time, we learn to process huge amounts of sensorial inputs. The combination of both these paradigms allows us to learn very quickly, using an almost infinite amount of data. Our future echolocating agents should be allowed to

learn the same way.

In order to do so, the environment we are using must be upgraded to mimic real-world acoustics. On the other hand, the agent we use must be able to learn how to process sound waves in real-time. Perhaps a Convolutional Neural Network that learns how to cluster sound waves based on their Doppler frequency shift could be an adequate first step. One however needs to figure out how these clusters can be mapped to an adequate action in the TD learning space. These questions are frankly very hard, and touch upon the roots of how we learn. Artificial Intelligence has come a long. It however still has a long way to go before mimicking mammalian brains, before mimicking the bat brain.

6. Conclusion

This project presents a simple way in which a learning agent could use echolocation to aid its path navigation capabilities. It bases itself upon the Foster spatial navigation model [1]. This classic model in neuroscience uses TD learning and an actor-critic architecture in order to train an agent to find a goal location in as few steps as possible. It also proposes a Coordinate system to speed up learning. We propose a simple alteration to both algorithms that incorporates echolocation in the learning process. The echolocation used is inspired by bats. This alteration significantly improves the learning process. It shows the need for learning algorithms to incorporate more information about the environment in the learning process. By doing so, reinforcement learning algorithms could start to mimic mammalian performance on certain tasks. This is absolutely crucial for tasks such as autonomous path navigation, that aim to be incorporated in the real world one day.

Acknowledgements

Thank you to Professors Alessandro Lazaric and Matteo Pirotta for allowing me to complete this project. Frankly, I am still trying to figure out how I got the idea to incorporate echolocation in path navigation. Alas, this was fun!

References

[1] David J Foster, Richard GM Morris, and Peter Dayan. A model of hippocampally dependent navigation, using the temporal difference learning rule. *Hippocampus*, 10(1):1–16, 2000.

[2] James R Hinman, Holger Dannenberg, Andrew S Alexander, and Michael E Hasselmo. Neural mechanisms of navigation involving interactions of cortical and subcortical structures. *Journal of neurophysiology*, 119(6):2007–2029, 2018.

[3] Michael Langer. Lecture notes on computational perception, April 2019.

[4] P Read Montague, Peter Dayan, and Terrence J Sejnowski. A framework for mesencephalic dopamine systems based on predictive hebbian learning. *Journal of neuroscience*, 16(5):1936–1947, 1996.

[5] R. G.M. Morris. Morris water maze. *Scholarpedia*, 3(8):6315, 2008. revision #91529.

[6] Yael Niv. Reinforcement learning in the brain. *Journal of Mathematical Psychology*, 53(3):139–154, 2009.

[7] Wolfram Schultz, Peter Dayan, and P Read Montague. A neural substrate of prediction and reward. *Science*, 275(5306):1593–1599, 1997.

[8] Richard S Sutton and Andrew G Barto. *Reinforcement learning: An introduction*. MIT press, 2018.

[9] Yossi Yovel and Stefan Greif. Bats–using sound to reveal cognition. *Field and Laboratory Methods in Animal Cognition: A Comparative Guide*, pages 31–59, 2018.

Appendix

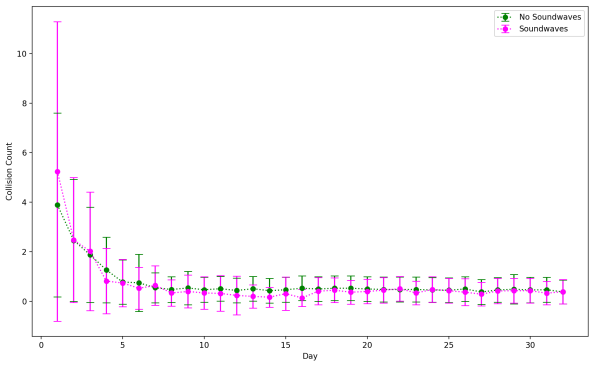

Figure 10. TD Learning Performance Comparison on the Regular Watermaze

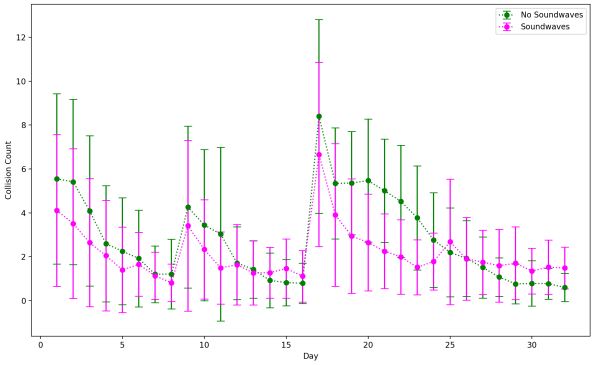

Figure 11. TD Learning Performance Comparison on the Obstacle Watermaze

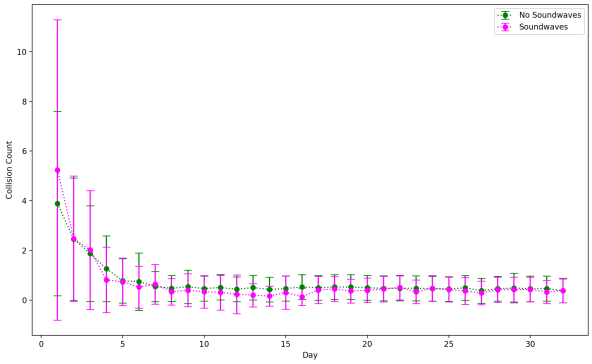

Figure 12. Coordinate System Performance Comparison on the Regular Watermaze

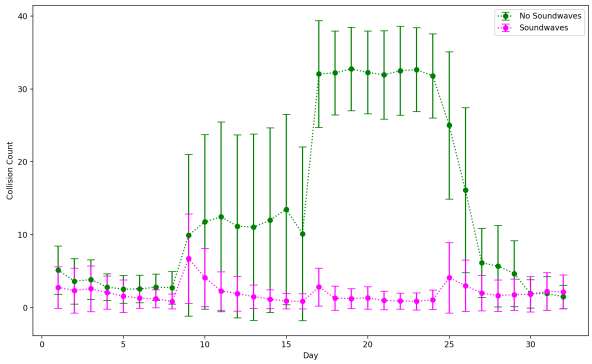

Figure 13. Coordinate System Performance Comparison on the Obstacle Watermaze