Optimizing Predator-Prey Behavior Through Q-Learning

Gregory Benton
CSCI 5622 Student
University of Colorado- Boulder
Boulder, CO 80309
gregory.benton@colorado.edu

Ian Char
CSCI 5622 Student
University of Colorado- Boulder
Boulder, CO 80309
ian.char@colorado.edu

Jonathon Lavington
CSCI 5622 Student
University of Colorado- Boulder
Boulder, CO 80309
jola2372@colorado.edu

Evan Sidrow
CSCI 5622 Student
University of Colorado- Boulder
Boulder, CO 80309
evsi8432@colorado.edu

1 Overview and motivation

In recent years, reinforcement learning has become relevant within a vast array of different sectors of science and technology. The primary goal of reinforcement learning is to iteratively update a model based upon what an “agent" has seen within similar situations. The more situations the agent is exposed to, the better it is able to adapt its strategy and maximize the reward that it can receive within an environment. The way this reward is distributed and how the agent updates its understanding of the environment are currently major topics of research. Using the paper "Reinforcement learning: A survey" [1] as a guide, we created an environment in which many of these hypotheses could be tested. Titling this environment "Grid World", we investigated how predator-prey agents interacted within a simple 2-D world. We first considered the scenario in which one predator attempts to capture one prey, then made the system more complicated by adding multiple agents of each type. Within these environments we investigated different parameter regimes, and by applying reinforcement learning to this simplified game environment, we were able to test the limits of the Q-learning algorithm and assess its benefits.

2 Grid world

To investigate these different game types, we developed a robust "Grid World" framework in which we were able to add different agents to the world, have the world advance in discrete time steps, and visualize the results. The code for this can be found on our GitHub repository: `https://github.com/g-benton/CSCI5622`. In summary, Grid World is a two dimensional environment that can be thought of as an $m \times n$ matrix. That is, it is a rectangular world that is comprised of $m * n$ different cells. Agents, such as predators and prey, occupy these cells and can move to the cell north, south, east, or west of their current location (as long as the destination is within the confines of the Grid World). If the predator agent moves to the same cell as a prey agent, the prey agent is "captured" and removed from the Grid World.

3 Basic algorithm

The driving force behind our trained agents was the Q-learning algorithm. Each agent had a Q-matrix “brain" with a row for every state and a column for every potential action (move east, west, north,

31st Conference on Neural Information Processing Systems (NIPS 2017), Long Beach, CA, USA.

south, or not at all). For example, suppose at time t the agent was in state s_t. With probability ϵ, the agent moved pseudo-randomly according to probabilities associated with values in row s_t of Q, and with probability $1 - \epsilon$, the agent performed the action associated with the largest value in row s_t of Q. After performing the action, Q updated according to equation (1) below.

$$Q(s_{t-1}, a_{t-1}) := (1 - \alpha)Q(s_{t-1}, a_{t-1}) + \alpha(r_t + \gamma \max_a Q(s_t, a)) \tag{1}$$

Where s_{t-1} was the previous state, a_{t-1} was the action taken to get to state s_t, α was the learning rate, γ was the discount rate, and r_t was the reward given for being in state s_t. For specific state spaces and rewards for each game, see the section 4 below. For descriptions and tuning of α, γ, and ϵ, see section 5.

4 Games

4.1 Training predator

4.1.1 Setup of the game

This was the original game type where we implemented Q-learning with a single predator agent and a single prey agent. The predator was trained via simulation using a cost function whereby it would receive a reward for successfully capturing the prey agent. The reward was invariant of the number of steps required to capture, and did not change from game to game. The state space used for the Q-matrix was comprised of a distance vector from the predator to the prey. Within this game type, the prey moved in a directed random walk, taking a step in the same direction as the last step with probability .35, in the opposite direction with probability .15, and to either side with probability .25. This was to ensure that the prey moved in a directed fashion, but was not completely predictable.

4.1.2 Outcome of training

Within this game type we found that the average performance of the predator converged after approximately 2000 iterations. In general, the predator showed signs of intelligent action, often cornering the prey to mitigate the possibility of escape. As a result of the simultaneous movement within the game, as well as the lack of diagonal movement, the predator tended to trail the prey until a boundary was reached. In figure 1 the dashed line represents the performance of a simple deterministic predator that always moves towards the prey.

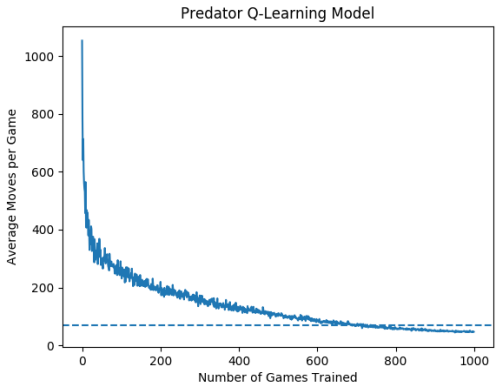

Figure 1: Time steps needed to capture the prey as the predator learns over games.

4.2 Training prey

4.2.1 Setup of game

With the Q-learning predator in place, we then developed a Q-learning prey agent. The fundamental difference between the initial predator agent and the prey agent is the inclusion of negative rewards.

Heuristically, we wanted the prey to avoid capture, which means seeking states in which it does not overlap with any predators and avoiding states in which it does overlap. Thus the rewards were +1 for moving to any state in which it survives, and -100 for being captured by the predator.

Next, we set a baseline performance using deterministic prey. This was accomplished by setting the deterministic prey and predator in a finite grid with a limited amount of moves (to halt cases where the prey is never captured). We determined that the prey agent was never caught by the predator, setting a baseline for the Q-learning prey. After approximately 1000-5000 rounds of training, the Q-learning prey is also able to avoid capture by the predator over the length of one game (1000 steps in the grid). What was most interesting about this result was not that the Q-learning agent was able to avoid capture (like the deterministic prey), but that it learned to avoid capture in a new and unexpected way.

4.2.2 Outcome of training

Shown on the left in figure 2 is the average number of steps until capture with the maximum number of steps (and the baseline result) as the dotted line. The plot on the right shows a snap-shot of the game with a Q-learning prey agent. The prey agent learns to get adjacent to the predator and continually swap positions with it. Since the agents move simultaneously, they are able to pass through one another, and the prey exploited this to continually swap places with the predator and never be caught.

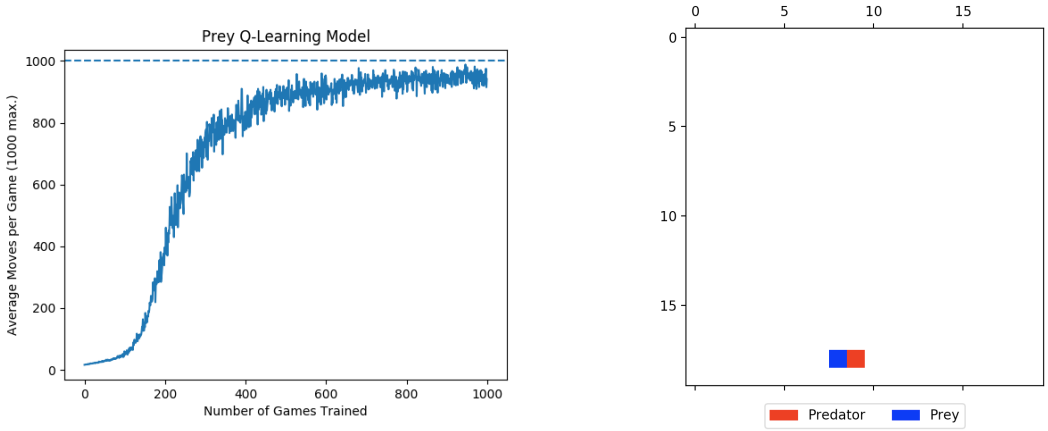

Figure 2: The average number of steps it takes until the prey gets captured (left) and the behavior the Q-learning prey exhibits (right)

4.3 Collaborative predators

4.3.1 Setup of game

We then moved on to the more complicated situation in which there were multiple predators and prey in the world. In particular, we explored the case in which there were two predators that learned via Q-learning (each with their own Q matrix) and chased three deterministic prey. Unlike the previous games, this game ended after T discrete time steps, and once prey were captured they were replaced at a random position. Additionally, we defined the score to be the total number of prey that the predators capture over the course of the T steps. Because the score is the combined number of prey captured by each predator, there is incentive for the predators to work together in order to maximize this score.

To make things even more interesting, the prey were equipped with a basic heuristic to help them avoid the predators. The basic outline of this altered prey's algorithm was to first find the closest predator, make a sorted list of ideal actions (move away from the predator in the axis in which the predator is closest), and pick the first action in which the prey will not run into another prey or wall. Implementing this made it much more difficult for the predator to capture prey.

In addition to altering the prey, the predator needed to be changed from previous games as well. The main reason for this was that classifying states based on the distance vector for each prey and each predator would have made the Q-matrix too large to store in memory. To fix this, we instead considered different "zones" that the predator could identify. Each zone was defined by the area outlined by several angles and radii (Figure 3). The predator could then identify how many predators and prey were in each zone; this information constituted the state space for this collaborative game.

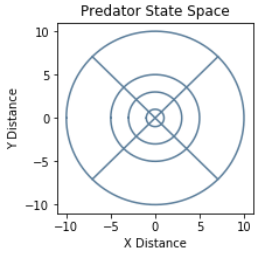

Figure 3: Representation of the predator's state space, where the predator is located at the center of the circles.

Another alteration to increase performance was the introduction of a sense of urgency for the predators. The speed in which predators catch prey played a bigger factor in this game since catching prey faster leads to a larger score. To do this, we introduced the parameter ρ, which is the fraction of the reward that was used as penalty associated with any state where the predator does not capture a prey. We found empirically that $\rho = 0.15$ gave good results by giving the correct balance between the predators having urgency and exhibiting herding behavior.

4.3.2 Outcome of Training

After training the predator agents for 10,000 games, each with 1,000 discrete time steps, we qualitatively observed collaborative behavior. In particular, we found that the predators often exhibited the strategy in which they lead prey into a corner and then approach them from opposite sides (Figure 4).

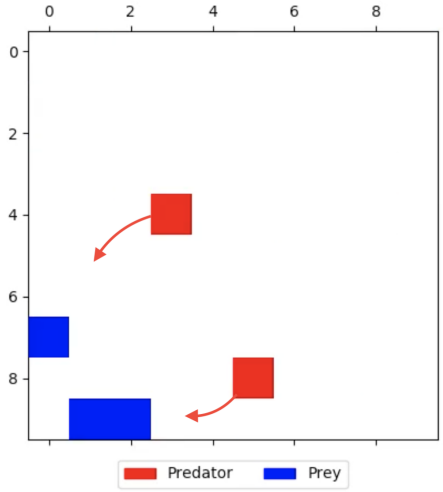

Figure 4: Trained predators exhibiting herding strategy.

To quantitatively evaluate the trained predators, we compared them to the heuristic-based predators, which simply try to minimize the distance to the closest prey. The predators were compared on a grid of size 5×5 and 10×10. Interestingly, it was found that the trained predators far exceeded the performance of the heuristic predators on the smaller grid, but the two groups performed about equally on the larger grid (Table 1). Upon inspection, it seems that many times the heuristic predators became caught in an infinite loop where they could not capture any prey in the 5×5 case. Thus, there is a major advantage to the trained predators, since they performed well regardless of environment.

Grid Size	Heuristic Predator (Baseline)	Trained Predator
5×5	15	189
10×10	94	93

Table 1: Comparison of average scores over (rounded to nearest integer) taken over 10,000 games, each having 1,000 discrete time steps, for heuristic and Q-learning predators.

5 Analysis of parameters

5.1 Description of parameters

5.1.1 α

α is the learning rate of the agents, and takes values between 0 and 1. In essence, Q is updated by a combination of the old Q and new information, and α determines the proportion of the old and new value (see equation (1) for details). If an agent does not receive the same reward (r) every time that it performs action a in a state s, then α for that state should be small, since it is important for the agent to "remember" past rewards. In the future, it is important to have α decay for each entry of Q after that entry is updated, since Q should change less as the predator learns more.

5.1.2 γ

γ is the "discount rate" of the agents, and takes values between 0 and 1. This parameter relates to how much the agent incorporates the quality of future states when updating its current state. For example, if $\gamma = 1$, then any reward will perfectly propagate back through the Q indefinitely. This could cause the agent to meander, since any path to success is equally rewarded, regardless of length. However, if $\gamma = 0$, then the agent will only look at the immediate reward and disregard any future information. This could cause the agent to lack any vision into the future and become entirely "greedy".

5.1.3 ϵ

ϵ is the probability that the agent will randomly explore the environment, rather than take the action associated with the maximum value in row s of Q. This is related to the classic "exploration vs exploitation" dilemma. We implemented 3 distinct values: an initial value, ϵ_0, a rate that ϵ decreases every time that a predator catches a prey (k), and a minimum value, ϵ_{min}.

5.2 Parameter tuning

For k and ϵ_{min}, we manually put in several values and saw $k = 0.999$ and $\epsilon_{min} = 0.01$ behaved optimally. We also looked at the convergence rate and performance for the simple trained predator game with various values of ϵ_0. Varying ϵ_0 did not effect the convergence rate or peak performance, so we arbitrarily choose $\epsilon_0 = 0.5$. For γ and α, we looked at the average moves per game after 1000 games for the simple trained predator game. This was done for several different values of γ and α. See figure (5) for details, but $\gamma = 0.9$ and $\alpha = 0.1$ were found to be optimal.

6 Results and conclusion

6.1 Outcome of games

In the simple one-on-one scenario, an "intelligent" prey (either learning or deterministic) always outperformed the predator. When the deterministic predator was placed against the deterministic prey, this occurred by moving about the grid in such a way that the predator never caught it. For the Q-learning prey, however, the agent uncovered a way to "win" the game that we had not thought of; the prey agent determined that if it moved adjacent to the predator and constantly moved towards the predator, the two agents would switch places indefinitely. We did not intend this to be a way to win the game, and it was an interesting result when found. Additionally, when comparing a Q-learning predator and a deterministic predator against a randomly moving prey, the Q-learning predator eventually outperformed the deterministic predator.

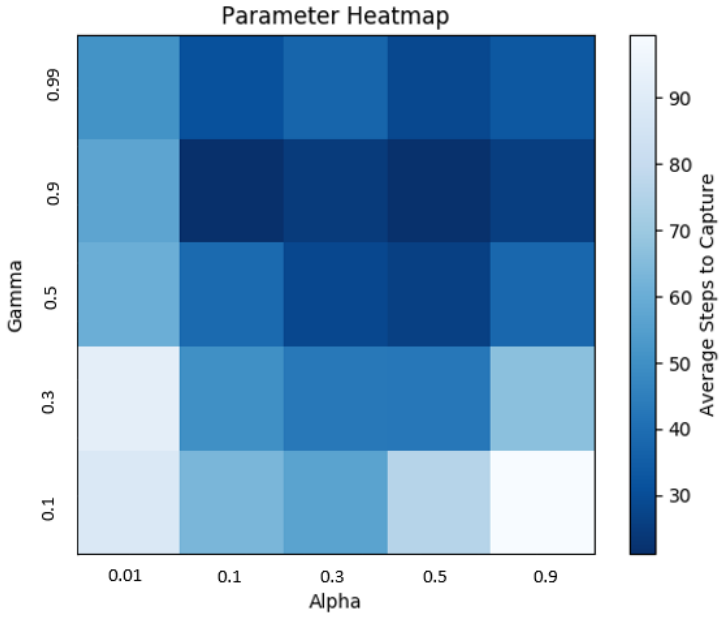

Figure 5: Predator performance after 1000 training instances. Note that the scale for γ and α is not linear.

In the scenario where multiple predators were used, the Q-learning agents outdid or matched the deterministic agents in all cases. The deterministic agents sought prey in a "naive" manner, just moving towards the closest target. The Q-learning agents sought prey in a much more organized way, using spontaneously arising collaboration to corner prey agents and achieve much higher scores in the game type.

6.2 Lessons learned and conclusion

In conclusion, we found that Q-learning is a powerful technique to create sophisticated agents. This becomes especially useful in cases where it is too difficult for computer scientists to derive the agent's optimal policy using human intuition. For example, in the collaborative game explored here, our best heuristic for predators in a 5×5 grid failed miserably, whereas the trained predators discovered a successful policy.

Even when agents with optimal policies can be coded by programmers, Q-learning can uncover alternate solutions. This was the case for the game in which an optimal prey was being trained, since both the baseline and trained prey essentially solved the game. However, the Q-learning prey did so with a new strategy. As such, we see that reinforcement learning can not only provide us with agents exhibiting sophisticated behavior, but also uncover knowledge about the environment or task for which it is trained.

7 Review of responsibilities

7.1 Gregory Benton

- Assisted with the initial setup and design of Grid World.
- Constructed the Q-learning prey agent and associated games.
- Built the visualization model to make animations and display the games.
- Contributed to the poster and final paper.
- Made tea.

7.2 Ian Char

- Designed/implemented most of Grid World.
- Set up collaborative predator game and tuned predators for performance.
- Contributed to the creation of poster and final paper.

7.3 Jonathon Lavington

- Programmed functions to pull information from different game types for visualizations.
- Created functions for visualizations of different parameters regimes and game/agent metrics.
- Created obstacle agent, and maze-world game type.
- Contributed to the creation of poster and final paper.

7.4 Evan Sidrow

- Conducted initial research on Q-learning and state-space reduction.
- Wrote the basic Q-learning algorithm.
- Specified the state space for each game.
- Tuned parameters (α, γ, and ϵ) for each game.
- Contributed to the poster and final paper.

References

[1] Kaelbling, Leslie Pack, Michael L. Littman, and Andrew W. Moore. "Reinforcement learning: A survey." *Journal of artificial intelligence research* 4 (1996): 237-285.

[2] Fern, Alan. *RL for Large State Spaces: Value Function Approximation.* web.engr.oregonstate.edu/ afern/classes/cs533/notes/rl-function-approximation.pdf.