Applying Q-Learning to Flappy Bird

Moritz Ebeling-Rump, Manfred Kao, Zachary Hervieux-Moore

***Abstract*— The field of machine learning is an interesting and relatively new area of research in artificial intelligence. In this paper, a special type of reinforcement learning, Q-Learning, was applied to the popular mobile game Flappy Bird. The Q-Learning algorithm was tested on two different environments. The original version and a simplified version. The maximum score achieved on the original version and simplified version were 169 and 28,851, respectively. The trade-off between run-time and accuracy was investigated. Using appropriate settings, the Q-Learning algorithm was proven to be successful with a relatively quick convergence time.**

I. INTRODUCTION

Q-Learning is form a reinforcement learning that does not require the agents to have prior knowledge of the environment dynamics [1]. The agents do not have access to the cost function or the transition probabilities. Reinforcement learning is a type of learning strategy where the agents are not told which actions to take. Instead, the agents discovers which action yields the highest reward from experimentation [2]. Q-Learning was first introduced by Watkins in 1989 [3] and it was not until 1992 that the convergence was shown by Watkins and Dayan [1].

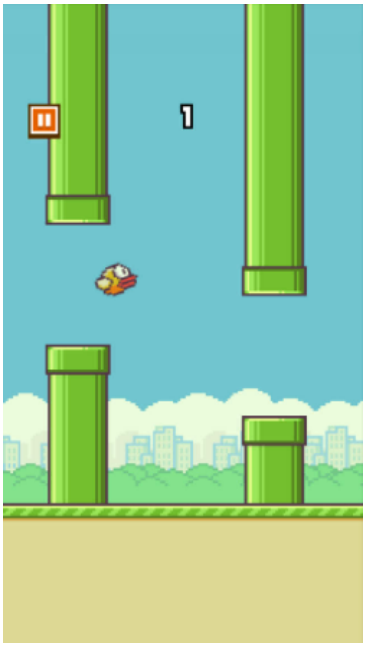

Fig. 1. The gameplay of Flappy Bird

Flappy Bird is a mobile game developed in 2013 by Dong Nyugen [4] and published by dotGears, a small independent game developer company based in Vietnam [5]. Flappy bird is a two-dimensional side-scrolling game, illustrated in Figure 1, featuring retro style graphics. The goal of the game is to direct the bird through a series of pipes. If the bird touches the floor or a pipe, then the bird will die and the game restarts. The only action that players are able to perform is to make the bird jump. Otherwise, the bird will fall due to gravity. Each pipe the bird successfully clears, the score is incremented. In the original game, the gap between the pipes are held constant but the height of the gap is uniformly distributed. In the simplified version that was considered, the height of the gap is held constant. The game is meant to test players' endurance as there is no end to the game.

II. THEORY

Watkins and Dayan first showed convergence of the Q-Learning algorithm in [1]. However, Tsitsiklis gives an alternate proof of convergence with stronger results in [6]. It is Tsitsiklis's results that will be shown here. To begin, define the standard form of a stochastic approximation algorithms.

$$x_i \coloneqq x_i + \alpha_i(F_i(x) - x_i + \omega_i) \tag{1}$$

Where $x = (x_1, x_2, ..., x_n) \in \mathbb{R}^n$, $F_1, F_2, ..., F_n$ are mappings from $\mathbb{R}^n \to \mathbb{R}$, ω_i is a noise term, and α_i is the step size. Assumptions on equations of this form are now listed. Note, in Tsitsiklis's paper, he allows for outdated information. For simplicity, it is assumed all information available to the controller is up to date which will simplify ths list. This is justified as the application considered has a controller that never uses outdated information.

Assumption 1:

(a) $x(0)$ is $\mathscr{F}(0)$-measurable.
(b) $\forall\ i$ and t, w_i is $\mathscr{F}(t+1)$-measurable.
(c) $\forall\ i$ and t, $\alpha_i(t)$ is $\mathscr{F}(t)$-measurable.
(d) $\forall\ i$ and t, we have $E[\omega_i(t)|\mathscr{F}(t)] = 0$.
(e) $\exists$ constants A and B such that

$$E[\omega_i^2(t)|\mathscr{F}(t)] \leq A + B \max_j \max_{\tau \leq t} |x_j(\tau)|^2, \forall \quad i, t$$

Assumption 1 is the typical restrictions imposed on stochastic systems. First, the present can be determined entirely by previous information. Second, the noise term has mean 0 and bounded variance.

Assumption 2:

(a) $\forall i$,

$$\sum_{t=0}^{\infty} \alpha_i(t) = \infty, \quad w.p.\,1$$

(b) $\exists\ C$ such that $\forall\ i$,

$$\sum_{t=0}^{\infty} \alpha_i(t)^2 \leq C, \quad w.p.\,1$$

Assumption 2 are restrictions on the step size to ensure that it approaches 0 at a slow enough speed to ensure convergence of the algorithm.

The following conventions are now used. If $x \leq y$ then $x_i \leq y_i\, \forall\, i$. The norm $\|\cdot\|_v$ is defined as:

$$\|x\|_v = \max_i \frac{|x_i|}{v_i}, \quad x \in \mathbb{R}^n$$

Assumption 3:

(a) The mapping F is monotone. That is, if $x \leq y$ then $F(x) \leq F(y)$
(b) The mapping F is continuous.
(c) The mapping F has a unique fixed point x^*.
(d) If $e \in \mathbb{R}^n$ is the vector whose components are all equal to 1, and $r > 0$, then,

$$F(X) - re \leq F(x - re) \leq F(x + re) \leq F(X) + re$$

It is not hard to imagine how Assumption 3 might be used to generate inequalities that are essential in a proof.

Assumption 4: $\exists$ a vector $x^* \in \mathbb{R}^n$, a positive vector v, and a $\beta \in [0, 1)$ such that,

$$\|F(x) - x^*\|_v \leq \beta \|x - x^*\|_v, \quad \forall x \in \mathbb{R}^n$$

It can be seen that the previous assumption provides the framework for a contraction argument (since $\beta \in [0, 1)$). This can be used to prove convergence to a stationary point.

Assumption 5: $\exists$ a positive vector v, and a $\beta \in [0, 1)$ and D such that,

$$\|F(x)\|_v \leq \beta \|x\|_v + D, \quad \forall x \in \mathbb{R}^n$$

Tsitsiklis goes on the prove the following three theorems.

Theorem 1: Let Assumptions 1,2, and 5 hold. Then, then sequence $x(t)$ in the stochastic approximation is bounded with probability 1.

Theorem 2: Let Assumptions 1,2, and 3 hold. Then, then sequence $x(t)$ in the stochastic approximation is bounded with probability 1.

Theorem 3: Let Assumptions 1,2, and 4 hold. Then, then sequence $x(t)$ in the stochastic approximation converges to x^* with probability 1.

From these theorems, it is clear that Theorem 3 is the strongest in the sense that we need the fewest assumptions to guarantee convergence. Also, Theorem 2 requires boundedness which is the result of Theorem 1. Thus, one can think of these as: to guarantee convergence, one needs Assumptions 1, 2, 4, and 5 or 1, 2, and 3.

With regards to Q-Learning, the way these theorems are important depends on writing $F(x)$ as follows:

$$F_{iu}(Q) = E[c_{iu}] + \beta \left[\min_{v \in U(s(t,u))} Q_{s(i,u),v} \right]$$

when $F(x)$ is written in this form, it can be shown that Assumptions 1 and 4 are met. Hence, by Theorem 3, we only need Assumption 2 for convergence to be guaranteed. This is an assumption on the step size α so it is very easy for the algorithm user to impose this restriction in practice.

For our application, we will rewrite this cost minimization as a reward maximization and use different notation as shorthand:

$$F\left(Q_t\left(s_t, a_t\right)\right) = \mathbb{E}\left[R_{t+1}\right] + \gamma \mathbb{E}\left[\max_a Q_t\left(s_{t+1}, a\right)\right]$$

We now start with the Q-Learning algorithm and work our way backwards to the stochastic approximation algorithm in equation 1.

$$\begin{aligned} Q_{t+1}\left(s_t, a_t\right) = & Q_t\left(s_t, a_t\right) + \alpha_t\left(s_t, a_t\right) \cdot \\ & \left\{R_{t+1} + \gamma \max_a Q_t\left(s_{t+1}, a\right) - Q_t\left(s_t, a_t\right)\right\} \\ & = Q_t\left(s_t, a_t\right) + \alpha_t\left(s_t, a_t\right) \cdot \\ & \{R_{t+1} + \mathbb{E}\left[R_{t+1}\right] - \mathbb{E}\left[R_{t+1}\right] \\ & + \gamma \mathbb{E}\left[\max_a Q_t\left(s_{t+1}, a\right)\right] - \mathbb{E}\left[\max_a Q_t\left(s_{t+1}, a\right) | I_t\right] \\ & + \max_a Q_t\left(s_{t+1}, a\right) - Q_t\left(s_t, a_t\right)\Big\}, \end{aligned}$$

where I_t denotes the information available to the controller at time t.

and

$$\begin{aligned} w_t\left(s_t, a_t\right) = & R_{t+1} - \mathbb{E}\left[R_{t+1}\right] + \max_a Q_t\left(s_{t+1}, a\right) \\ & - \mathbb{E}\left[\max_a Q_t\left(s_{t+1}, a\right) | I_t\right]. \end{aligned}$$

Rewriting the expression for Q we get

$$\begin{aligned} Q_{t+1}\left(s_t, a_t\right) = & Q_t\left(s_t, a_t\right) + \alpha_t\left(s_t, a_t\right) \cdot \\ & \left(F\left(Q_t\left(s_t, a_t\right)\right) - Q_t\left(s_t, a_t\right) + w_t\left(s_t, a_t\right)\right) \end{aligned}$$

Q is now in the desired form! Thus, we only need to impose the restictions on $\alpha(t)$ to guarantee convergence of Q.

The last case we wish to consider is when the problem is undiscounted ($\beta = 1$). For this, one further assumption is needed.

Assumption 6:

(a) $\exists$ at least one proper stationary policy
(b) Every improper stationary policy yields infinite expected cost for at least one initial state

Where a proper policy is defined to be a stationary policy that the probability of going to an absorbing state with 0 reward converges to 1. Otherwise, it is improper. With this, we get the last theorem:

Theorem 4: The Q-Learning algorithm converges with probability 1 if we impose Assumption 2 and:

(a) $\beta < 1$,
(b) $\beta = 1, Q_{iu}(0) = 0$, and all policies are proper, or
(c) $\beta = 1, Q_{iu}(0) = 0$, Assumption 6 holds, and $Q(t)$ is guaranteed to be bounded with probability 1.

III. IMPLEMENTATION

The key to a successful implementation of Q-Learning is modeling the problem efficiently. As long as the state space is finite, Q-Learning will converge. In practical applications, we are looking to obtain results in a limited time frame. If the state space is very large, the algorithm will take too long to converge. On the other hand, if the state space is very small, accuracy will be lost. Thus, there is a trade-off between run-time and accuracy.
The dimension of the state space will be kept to a minimum by only considering three attributes:

- X distance from the next pipe
- Y distance from the next pipe
- Life of the bird

A visual representation of the first two attributes are shown in Figure 2

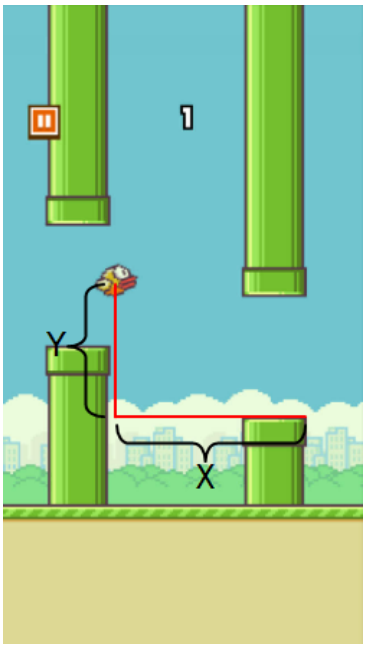

Fig. 2. The first two attributes: X distance & Y distance

Only integers are considered to ensure a finite state space. The distance in X direction is bounded by 0 from below and by 300 from above. The Y distance ranges from -200 to 200. The state of the bird is characterized by dead as 0 and the alive as 1. Thus the state space is defined as:

$$\mathbb{X} = [0, 300] \times [-200, 200] \times \{0, 1\}.$$

At every point in time, there are two possible actions - jump or don't jump. Thus the action space is defines as:

$$\mathbb{U} = \{0, 1\}.$$

After the previous action has been applied, the resulting state is analyzed. A reward function is invoked, which reinforces or punishes the previous action. If the bird survived, the previous action is seen as positive and is being reinforced. However, if the bird died, we apply the punishment. The exact values of the reward function are not as important as long as reinforcement is positive and punishment is negative. We weigh the benefit of survival lower than the punishment of death, as the goal is to avoid death.

$$R_{t+1} = \begin{cases} 15, & \text{if bird alive} \\ -1000, & \text{if bird dead} \end{cases}$$

The learning rate determines to what extent new information overrides old information. The following learning was chosen:

$$\alpha_t(s_t, a_t) = \frac{1}{1 + \sum_{k=0}^{t} 1_{\{s_k = s_t, a_k = a_t\}}}$$

The discount factor determines the importance of future rewards. This value is given below.

$$\gamma = 1.$$

These are the essential definitions that are needed to implement the Q-Learning algorithm.
Now the Q-Learning pseudocode that we used in our implementation is shown below.

1) Initialize Q matrix
2) Repeat
   a) Determine the action, a, in state, s, based on Q matrix
   b) Take the action, a, and observe the outcome state, s, and reward, r
   c) Update Q matrix based on Equation 2

$$Q_{t+1}(s_t, a_t) = Q_t(s_t, a_t) + \alpha_t(s_t, a_t) \cdot \left(R_{t+1} + \gamma \max_a Q_t(s_{t+1}, a) - Q_t(s_t, a_t)\right) \quad (2)$$

IV. DISCUSSION

When starting the Q-Learning algorithm, the Q matrix is initialized to zero. The Q matrix can be seen as the brain of the artificial intelligence. At the beginning, it does not have any knowledge about its environment. When applying Equation 2 for the first time step, the maximum is taken between 0 and 0. In other words: faced with the decision of jumping or not jumping, both actions have the same Q values - a tie occurs. We decided to choose the action "don't jump" as a tie breaker because it is the more human-like option.
The bird was expected to fall to the ground in the first couple of trials, but learn to avoid the ground fairly quickly since falling to the ground is not a good option.
The Q matrix described before is of size 300 x 400 x 2. Even though that doesn't seem that big, it is still a total of 240,000 different states. That means that every time the bird is positioned slightly different relative to the pipe, it does not have any knowledge. However, the algorithm still converges. It just takes a while to fill the Q matrix. That is why the state space was discretized further for states where the bird is far away from the initial pipe.
The first intelligent behavior is the bird learns that falling to the ground is a bad idea. Note that the bird doesn't have

information about what "ground" is or even where it is located. The only available information is the distance to the next pipe and the current life status.

The next stage in the bird's learning process is that it will crash into the lower pipe until it learns that this leads to a negative reward. With the tie breaker being "don't jump", it will fall whenever it encounters a new state. After getting feedback, it will realize that falling was not a good move in the prior situation. If the bird is located in the same position relative to a pipe again it will use this information and jump instead. This propagation continues leading to the bird eventually clearing the first pipe.

Another problem arouse: even after flying through the gap the bird would eventually die. The bird learned that flying through the pipe will lead to death. Due to the back propagation, the Q matrix had negative entries for states close to the gap. Following the algorithm, the states above the gap with Q values of zero would get explored. For all the AI knows, there could be a better option than flying through the gap. It continues to go through all other states trying to find an alternative before realizing that flying through the gap was the best way.

It was observed that the speed of the bird makes a big difference. The game engine works with gravity. The longer the bird is falling, the faster it gets. This influences the choice of the optimal policy. We could include the speed of the bird in the state space, but that would add another dimension to the state space, which would slow down the learning process tremendously.

That's why a different approach was taken. The learning rate was lowered with regard to how often a certain state was visited before.

The dynamic learning rate takes care of both problems described above. The AI learns that flying closely above the pipe leads to positive benefits. Even if it does eventually crash, it doesn't disregard this information. The case of the bird crashing because of missing information regarding speed is rather seldom. The adjusted learning rate makes sure that these improbable events do not have too big of an impact on the Q matrix.

The bird successfully learned how to play the game using Q-Learning without any prior knowledge of the game dynamics. A high score of 169 and 28,851 was achieved on the original and simplified game, respectively.

Figure 3 shows the learning rate of the bird by comparing the number of trials and their respective scores.

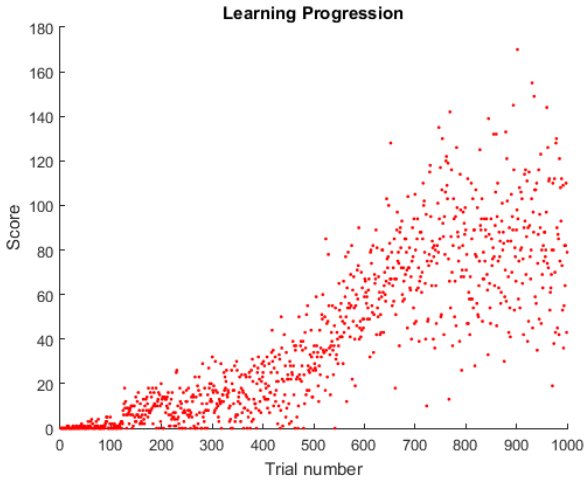

Fig. 3. The score plotted with the number of trials of the AI

The convergence of the algorithm is ensured due to the choice of a suitable $\alpha(t)$ and appropriateness of the state space since the discounted factor $\gamma = 1$ was chosen. We know that Assumption 2 on α is satisfied due to:

$$\sum_{t=0}^{\infty} \alpha_t(s_t, a_t) = \sum_{t=0}^{\infty} \frac{1}{1 + \sum_{k=0}^{t} 1_{\{s_k = s_t, a_k = a_t\}}} = \sum_{n=1}^{\infty} \frac{1}{n} = \infty$$

Likewise,

$$\sum_{t=0}^{\infty} \alpha_t(s_t, a_t)^2 = \sum_{n=1}^{\infty} \frac{1}{n^2} = 2$$

Since $\gamma = 1$ is used, it needs to satisfy the conditions of Theorem 4, in particular, part b or c. We initialize the Q matrix to 0, which is a requirement for both. Next, we impose additional structure on the Q. We bound Q from above by 10,000. That is, reward becomes 0 once the Q matrix entry becomes sufficiently large. Also, it is known *a priori* that the game is solvable. That is, there is an optimal policy that will guarantee the bird's survival for eternity. This is known by the construction of the game. The code does not allow for a sequence of pipes that will ensure death. Hence, we have two absorbing states. One being the death of the bird, as the bird cannot revive. The other is the collection of states that make the bird live forever. Since we bound the Q matrix, we have a proper stationary policy that has 0 reward and all other improper policies (those leading to death) have negative infinite expected reward. Thus, we satisfy the conditions for part c of Theorem 4 and, by consequence, have convergence of the Q matrix.

Although convergence is guaranteed, in practice, one does not have the luxury of running the algorithm forever. Thus, the greatest critique of this algorithm is that convergence may take a long time, which is undesirable. For instance, if the controller learns a good behavior, but sub-optimal, it will

continue to perform this action even though better actions exist until those actions are explored. In this application, the majority of the states in the state space are undesirable. If the controller misses the small window of desirable states, it might have to explore the entire state space before returning to that small window which will greatly increase the time of convergence.

V. POSSIBLE IMPROVEMENTS

If given more time, many ideas could have been implemented to improve the Q-Learning algorithm. Some of these ideas include techniques for faster convergence and methods to strengthen the survivability rate of the bird.

In the project, the graphics was displaying throughout the learning process of the bird. This significantly slows down the algorithm. If the graphics were to be removed from the code, the optimal policy can be achieved in a significantly reduced time frame.

Only one bird was used to find the optimal policy throughout the project. If multiple birds were simultaneously learning and they collectively updated the matrix Q, then this will greatly reduce the time needed to fill the Q matrix. This will result in the birds converging to the optimal policy much quicker.

When the bird dies, the game restarts with the bird at the same position every time. If a random starting position were to be implemented, then the bird will be able to explore the state space more efficiently. This may reduce the time it takes for the bird to converge to the optimal policy.

To improve the bird's survivability, a possible solution is to add another dimension to the state space. The bird's state is highly dependent on the velocity of the bird. If velocity is included in the state space, then the transition probability would be deterministic. This would allow the bird to learn precisely what the next state is going to be given its current state and action.

After the bird has converged to the optimal policy, it was noticed that there was only one case that lead to the death of the bird. This case was when the height of the gap of the current pipe was low to the ground while the height of the gap of the next pipe was near the ceiling. This case did not give the bird enough room to make it to the next pipe. A solution to this would be to include the position of the next pipe in the state space. This would allow the bird to plan ahead which in turn would increase its survivability rate.

VI. CONCLUSION

The big advantage of the Q-Learning algorithm is that it converges without knowledge of the system. However, the data given to the algorithm highly influences the run-time. The key to receiving meaningful results in a limited time-frame is the choice of an efficient state space. Following this observation, appropriate spaces and parameters were chosen. Even with a short learning time, the artificial intelligence exceeded expectations by performing better than a beginner human player. The Q-Learning algorithm can be improved upon by implementing the improvements outlined in Section V.

APPENDIX

The Matlab code for the Q-L earning algorithm is shown below. Note that only the relevant part of the code is included.

```
x_metric = round(Tubes.ScreenX...
    (Tubes.FrontP)-Bird.ScreenPos(1)...
    +5+16+4);
y_metric = round(177 - (Tubes.VOffset...
    (Tubes.FrontP)-1) -Bird.ScreenPos(2)...
    +5-10);
%If the x metric is negative
%(passed the pipe) look at the next pipe
if x_metric < 0
    x_metric = round(Tubes.ScreenX(mod...
        (Tubes.FrontP,3)+1) -...
        Bird.ScreenPos(1)...
        +5+16+4);
    y_metric = round(177 - ...
        (Tubes.VOffset(mod...
        (Tubes.FrontP,3)+1)...
        -1)-Bird.ScreenPos(2)+5-10);
end

%Find indices
for k=1:length(x)
    if x_metric<x(k)
        x_ind=k;
        break;
    end
end

for k=1:length(y)
    if y_metric<y(k)
        y_ind=k;
        break;
    end
end

x_ind_o=1;
for k=1:length(x)
    if x_metric_old<x(k)
        x_ind_o=k;
        break;
    end
end
y_ind_o=1;
for k=1:length(y)
    if y_metric_old<y(k)
        y_ind_o=k;
        break;
    end
end
```

```
%Calculate alpha
stateCount(x_ind_o,y_ind_o)=...
    stateCount(x_ind_o,y_ind_o)+1;
alpha = 1/(1+stateCount);
action_index=action+1;
if deathFlag==true
    R = -1000;
else
    R = 15;
end
%Q-learning algorithm
Q_temp = ...
    Q(x_ind_o,y_ind_o,action_index)+...
    alpha*(R*max(Q(x_ind,y_ind,:)) -...
    Q(x_ind_o,y_ind_o,action_index));

if Q_temp <= 10000
    Q(x_ind_o,y_ind_o,action_index)...
        = Q_temp;
end
deathFlag=false;

%Take action based on Q matrix
if Q(x_ind,y_ind,1)<Q(x_ind,y_ind,2)
    action=1;
    FlyKeyStatus=true;
else
    action=0;
end

x_metric_old=x_metric;
y_metric_old=y_metric;
```

REFERENCES

[1] C. J. Watkins and P. Dayan, "Q-learning," *Machine learning*, vol. 8, no. 3-4, pp. 279–292, 1992.

[2] R. S. Sutton and A. G. Barto, *Reinforcement learning: An introduction*. MIT press, 1998.

[3] C. J. C. H. Watkins, *Learning from delayed rewards*. PhD thesis, University of Cambridge England, 1989.

[4] R. Williams, "What is flappy bird? the game taking the app store by storm," jan 2014.

[5] M. Bertha, "Everything you need to know about your new favorite cell phone game, 'flappy bird'," jan 2014.

[6] J. N. Tsitsiklis, "Asynchronous stochastic approximation and q-learning," *Machine Learning*, vol. 16, no. 3, pp. 185–202, 1994.