**TRƯỜNG ĐẠI HỌC CÔNG NGHIỆP THỰC PHẨM TP. HỒ CHÍ MINH**



**TIỂU LUẬN**

**ĐỀ TÀI: Áp dụng thuật toán Q Learning vào flappy birth**

Giảng viên hướng dẫn: **Nguyễn Thị Hoài Thu**

Nhóm sinh viên thực hiện: Nhóm 11

Phạm Minh Nhựt - MSSV: 2001216012

**THÀNH PHỐ HỒ CHÍ MINH, THÁNG 9 NĂM 2023**

**Mục lục**

[1. Giới thiệu về Reinforcement Learning 2](#_Toc5648)

[2. Giới thiệu cơ bản về thuật toán. 2](#_Toc8973)

[2.1. Phương trình Bellman (Bellman Equation) 2](#_Toc6776)

[2.2. Q learning 4](#_Toc12802)

[2.3. Điểm khác biệt giữa Q learning và Bellman Equation. 5](#_Toc10032)

[3. Giới thiệu trò chơi. 8](#_Toc13487)

[3.1. Vòng đời. 8](#_Toc9164)

[3.2. Cách chơi. 8](#_Toc13211)

[3.3. Mục tiêu. 8](#_Toc16618)

[3.4. Công cụ sử dụng để build game 9](#_Toc9790)

[3.5. Giới thiệu các thuộc tính Q Learning trong bài toán 9](#_Toc17452)

[3.6. Hình ảnh 1 lần chạy tiêu chuẩn 11](#_Toc24464)

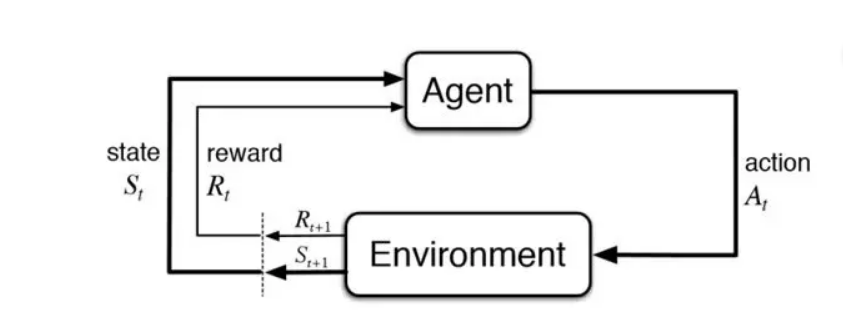
[4. Yêu cầu cấu hình chạy phần mềm 11](#_Toc20744)

[Tài liệu tham khảo 12](#_Toc2086)

**Nội dung : Flappy Bird Chim Bay Không Mỏi.**

# Giới thiệu về Reinforcement Learning

 Chương trình sẽ học cách thực hiện nhiệm vụ bằng cách tương tác với môi trường thông qua các hành động và dựa trên phần thưởng qua từng hành động mà đưa ra lựa chọn tối ưu.



* Environment (Môi trường): là không gian mà máy tương tác
* Agent (Máy): là chủ thể tương tác với môi trường qua hành động
* Policy (Chiến thuật): là chiến thuật mà máy sử dụng để đưa ra hành động
* State (Trạng thái): mô tả trạng thái hiện tại của máy
* Reward (Phần thưởng): phần thưởng từ môi trường tương ứng với hành động được thực hiện
* Action (Hành động): là những gì máy có thể thực hiện

Chúng ta sẽ bắt đầu từ state **S(t)**, tại trạng thái này, máy(agent) sẽ dựa trên chiến thuật mà ta đã vạch ra sẵn để đưa ra hành động (action) **A(t)**tương tác với môi trường .  Môi trường sau khi quan sát hành động sẽ chuyển đổi sang trạng thái tiếp theo**S(t+1)**đối với máy(agent) , và với mỗi lần tương tác với môi trường sẽ trả về một kết quả (phần thưởng) **R(t)** tương ứng. Agent sẽ lặp đi lặp lại qui trình này cho đến khi tìm được chuỗi những hành động tối đa hóa phần thưởng được nhận.

# Giới thiệu cơ bản về thuật toán.

## Phương trình Bellman ([Bellman Equation](https://en.wikipedia.org/wiki/Bellman_equation" \t "https://viblo.asia/p/_blank))

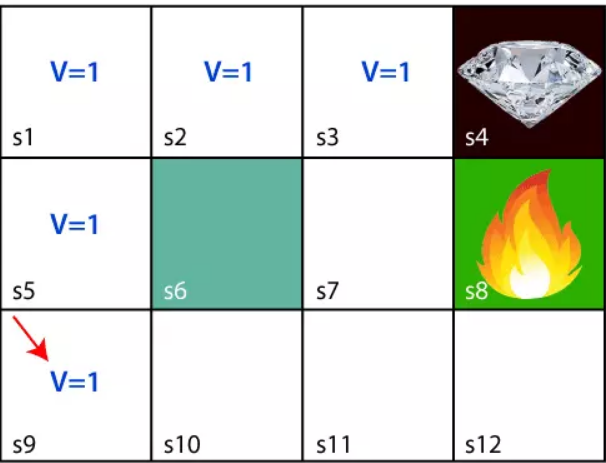
Phương trình Bellman được nhà toán học Richard Ernest Bellman giới thiệu vào năm 1953 và do đó nó được gọi là phương trình Bellman. Nó được liên kết với lập trình động và được sử dụng để tính toán các giá trị của bài toán quyết định tại một điểm nhất định bằng cách bao gồm các giá trị của các trạng thái trước đó.

(s’))]

**Trong đó**

* V(s)= giá trị được tính tại một điểm cụ thể.
* R(s,a) = Phần thưởng ở trạng thái cụ thể s bằng cách thực hiện một hành động.
* γ = Hệ số chiết khấu (do mình quy định)
* V(s`) = Giá trị ở trạng thái trước đó.

Giả sử chúng ta có bài toán về Robot tìm kim cương , giả sử chúng ta đứng tại vị trí s9 , lúc này mọi thứ giá trị đều ngang nhau là V = 1 , thì lúc này khi đó Robot sẽ có 2 hướng đi với các giá trị như nhau. Vậy chúng ta làm thế nào để đảm bảo được agent sẽ thực hiện đúng những hành động tối ưu mà chúng ta muốn?



Chúng ta có các bước thực hiện :

**- Đầu tiên**

V(s3) = max [R(s,a) + γV(s`)], ở đây V(s')= 0.

 R(s, a)= 1, vì có phần thưởng ở trạng thái này.

V(s3)= max[R(s,a)]=> V(s3)= max[1]=>**V(s3)**= 1.

**- Khối tiếp theo**

V(s2) = max [R(s,a) + γV(s`)], γ= 0.9(do mình quy ước), V(s')= 1

R(s, a)= 0, vì không có phần thưởng ở trạng thái này.

V(s2)= max[0.9(1)]=> V(s)= max[0.9]=> **V(s2)** =0.9

**- Khối tiếp theo**

V(s1) = max [R(s,a) + γV(s`)], γ= 0.9, V(s')= 0.9

R(s, a)= 0.

V(s1)= max[0.9(0.9)]=> V(s3)= max[0.81]=> **V(s1)** =0.81

**- Khối tiếp theo**

V(s5) = max [R(s,a) + γV(s`)], γ= 0.9, V(s')= 0.81

R(s, a)= 0

V(s5)= max[0.9(0.81)]=> V(s5)= max[0.81]=> **V(s5)** =0.73

...................................

(áp dụng công thức cho tất cả các khối còn lại)

|  |  |  |  |
| --- | --- | --- | --- |
| V = 0.81  (S1) | V = 0.9  (S2) | V = 1  (S3) | Phần thưởng |
| V = 0.73  (S5) | Đường cục | V = 0.9  (S7) | Bẫy |
| V = 0.66  (S9) | V = 0.73  (S10) | V = 0.81  (S11) | V = 0.73  (S12) |

## Q learning

Q -learning được giới thiệu bởi Watkins vào năm 1989. Một bằng chứng hội tụ đã được trình bày bởi Watkins và Dayan vào năm 1992. Một bằng chứng toán học chi tiết hơn bởi Tsitsiklis vào năm 1994, và bởi Bertsekas và Tsitsiklis trong cuốn sách Lập trình động học Neuro năm 1996 của họ

Q-learning là một thuật toán học tăng cường không có mô hình phổ biến dựa trên phương trình Bellman.

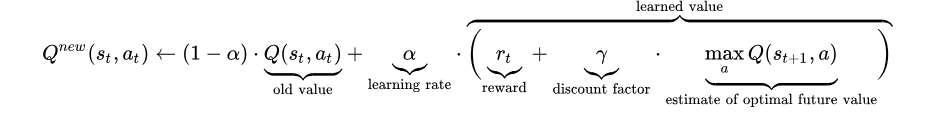
Sử dụng : Q Learning của phương pháp kĩ thuật học tăng cường (thường được sử dụng trong tìm đường đi hay xe tự động lái)

- Nó tác động với môi trường , dựa theo phần thưởng để trả về kết quả tối ưu nhất.

## Điểm khác biệt giữa Q learning và Bellman Equation.

Thay vì dựa trên giá trị của các state **V(s)** mà đưa ra quyết định về hành động thì Q-Learning tập trung hơn vào việc đánh giá chất lượng của một hành động **Q(s, a)**

**Không phụ thuộc vào mô hình**: Phương trình Bellman yêu cầu biết hoặc ước lượng được mô hình của môi trường. Cần biết chính xác các xác suất chuyển trạng thái và phần thưởng tương ứng để có thể áp dụng phương trình Bellman. Trái lại, Q-learning không yêu cầu thông tin về mô hình và có thể học từ dữ liệu trực tiếp.



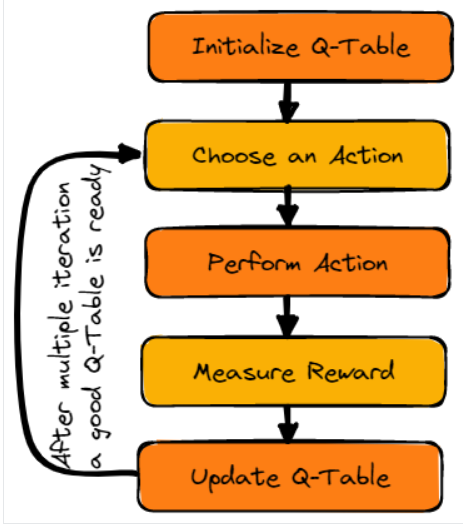
Q(s,a) là giá trị của hàm Q cho hành động a từ trạng thái s.

α là tốc độ học.

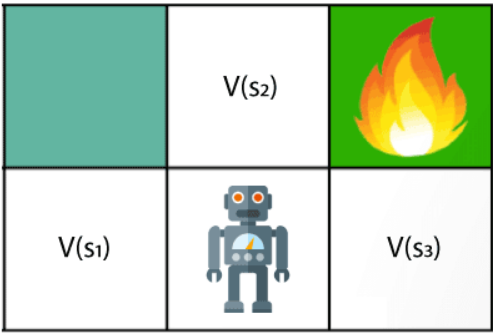
R(s,a) là phần thưởng nhận được khi thực hiện hành động a từ trạng thái s.

γ là hệ số giảm của phần thưởng trong tương lai.

s′là trạng thái tiếp theo sau khi thực hiện hành động a từ trạng thái s.



Giả sử chúng ta có



- Ban đầu, khởi tạo ma trận Q với giá trị ban đầu.

s0: [1, 0, 1, 1] (tương ứng với hành động đi lên, đi xuống, đi qua trái, đi qua phải )

s1 [ 1, 0, 0,1]

s2: [1, 1, 0, 0]

s3: [1, 0, 1, 0]

- Áp dụng công thức Q-learning từ dữ liệu:

**Các ô đi được**

Q(s1,A1) = (1 - 0.5) x 1 +1 x (0 + 0.9 x max(Q(s1,a’)))

Q(s1,A1) = 0.5 + 1 x (0 + 0.9 x 1)

Q(s1,A1) = 1.4

**Ô không đi được**

Q(s1,A1) = (1 - 0.5) x 0 +1 x (0 + 0.9 x max(Q(s1,a’)))

Q(s1,A1) = 0 + 1 x (0 + 0.9 x 1)

Q(s1,A1) = 0.9

Tương tự với s1 , s2 , s3

Dưới đây là ví dụ cụ thể :

Với hành động "đi lên" (action 1), giá trị Q đã cho là 0.

Với hành động "đi xuống" (action 2), giá trị Q đã cho là 0.

Với hành động "đi qua trái" (action 3), giá trị Q đã cho là 0.

Với hành động "đi qua phải" (action 4), giá trị Q đã cho là 1.

**Q(s1,A1) đi lên**

= 0.9 , 1

Với hành động đi lên (action 1) ta tính được giá trị như sau

Q(s1,A1) = (1 - 0.5) x 0 +1 x (0 + 0.9 x max(Q(s1,a’)))

Q(s1,A1) = 0 + 1 x (0 + 0.9 x 1)

Q(s1,A1) = 0.9

**Q(s1,A2) đi xuống**

= 0.9 , 1

Với hành động đi lên (action 2) ta tính được giá trị như sau

Q(s1,A1) = (1 - 0.5) x 0 +1 x (0 + 0.9 x max(Q(s1,a’)))

Q(s1,A1) = 0 + 1 x (0 + 0.9 x 1)

Q(s1,A1) = 0.9

**Q(s1,A3) đi trái**

= 0.9 , 1

Với hành động đi lên (action 3) ta tính được giá trị như sau

Q(s1,A1) = (1 - 0.5) x 0 +1 x (0 + 0.9 x max(Q(s1,a’)))

Q(s1,A1) = 0 + 1 x (0 + 0.9 x 1)

Q(s1,A1) = 0.9

**Q(s1,A4) đi phải**

= 0.9 , 0.5

Với hành động đi lên (action 4) ta tính được giá trị như sau

Q(s1,A1) = (1 - 0.5) x 1 +1 x (0 + 0.9 x max(Q(s1,a’)))

Q(s1,A1) = 0.5 + 1 x (0 + 0.9 x 1)

Q(s1,A1) = 1.4

........ (tương tự với các trường hợp còn lại)

Ngoại từ Bẫy

**Q(s3,A1) đi lên**

Với hành động đi lên (action 1) ta tính được giá trị như sau

Q(s1,A1) = (1 - 0.5) x 1 +1 x (-100 + 0.9 x max(Q(s1,a’)))

Q(s1,A1) = 0.5 + 1 x (-100 + 0.9 x 1)

Q(s1,A1) = -98.6

Giá trị cực tệ so với các trường hợp khác

Hệ thống tiến hành học tập , để tránh sai lầm này.

# Giới thiệu trò chơi.

## Vòng đời.

Flappy Bird  là một [trò chơi điện tử trên điện thoại](https://vi.wikipedia.org/wiki/Tr%C3%B2_ch%C6%A1i_%C4%91i%E1%BB%87n_t%E1%BB%AD" \o "Trò chơi điện tử) do [Nguyễn Hà Đông](https://vi.wikipedia.org/wiki/Nguy%E1%BB%85n_H%C3%A0_%C4%90%C3%B4ng" \o "Nguyễn Hà Đông), một lập trình viên ở [Hà Nội](https://vi.wikipedia.org/wiki/H%C3%A0_N%E1%BB%99i" \o "Hà Nội) vào năm 2013 và đã gây sốt cộng đồng mạng , nhưng đáng tiếc trò chơi đã bị gỡ xuống bởi chính tác giả vào ngày 10 tháng 2 năm 2014 do anh cảm thấy tội lỗi do những tác nhân tiêu cực mà nó gây ra .

## Cách chơi.

Mục đích là giữ cho flappy bird sống lâu nhất có thể , nó sẽ liên tục chịu tác dụng của lực hút trái đất kéo xuống và phải khéo léo vượt qua các đường ống xuất hiện trước mặt một cách ngẫu nhiên trò chơi sẽ kết thúc khi ta đến được điểm đích (999) hoặc khi chúng ta chạm vào ống nước hoặc rớt xuống đất.

## Mục tiêu.

Chúng ta : là phải làm sao để truyền được các giá trị môi trường (pixcel) , background , ống nước để flappy có thể vượt qua được ống một các dễ dàng.

## Công cụ sử dụng để build game

python 3 (là có thể chạy)

Pygame

Pytorch (sử lý hình ảnh đầu vào)

numpy

## Giới thiệu các thuộc tính Q Learning trong bài toán

Goat (đích) -> vượt qua các ống đến đích cuối cùng

Enviroment -> môi trường , ống , flappy birth

Agent -> máy -> (train cách flappy bird có thế vượt qua được các ống)

Static -> trạng thái -> ống cao thấp , dài ngắn khác nhau

Acction -> bay lên hay không bay (chịu tác động của lực hút)

Reward -> hướng đến hành động trả về kết quả cao -> cụ thể là flappy bird qua được ống trót lọt (1) đụng ống là (0)

Chúng ta đang mong muốn đạt được kết quả trong một khung thời gian giới hạn.

Nếu không gian trạng thái rất lớn thì thuật toán cũng sẽ mất

hội tụ lâu. Mặt khác, nếu không gian trạng thái là

rất nhỏ, độ chính xác sẽ bị mất. Vì vậy, có sự đánh đổi

giữa thời gian chạy và độ chính xác.

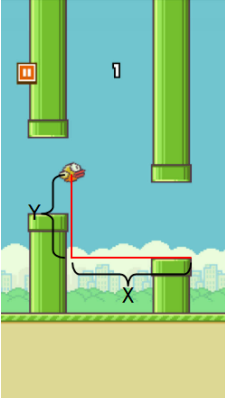
Kích thước của không gian trạng thái sẽ được giữ ở mức tối thiểu

bằng cách chỉ xem xét ba thuộc tính:

• Khoảng cách X từ ống tiếp theo

• Khoảng cách Y từ đường ống tiếp theo

• Trạng thái của con chim (chết 0 , sống 1)



Ta giả định khoảng cách từ X từ ống tiếp theo là 0 -> 300

Y từ ống tiếp theo là -200 -> 200

Trạng thái của chim có 2 trạng thái sống hoặc chết (1 , 0)

**Lúc này thu được**

X = [0, 300] × [−200, 200] × {0, 1} × 2

= 240000

Cùng với đó chúng ta có trạng thái đưa ra quyết định là nhảy hoặc không nhảy

U = {0,1}

Sau khi thực hiện các hành động thức đó , trạng thái phần thưởng (reward) sẽ được trả về tương ứng với lựa chọn hành động thực hiện trước đó . Tùy vào kết quả Một chức năng khen thưởng được gọi ra để củng cố hoặc trừng phạt hành động trước đó. Nếu con chim sống sót,hành động trước đó được coi là tích cực và đang được củng cố.Tuy nhiên, nếu con chim chết thì chúng ta áp dụng hình phạt.

Các giá trị chính xác của hàm thưởng không quan trọng bằng việc tăng cường tích cực và hình phạt tiêu cực.Chúng tôi đánh giá lợi ích của sự sống thấp hơn so với hình phạt của cái chết, vì mục tiêu là tránh cái chết. Ví dụ : bạn làm bài tập thì bạn sẽ được kẹo , còn không thì phải chép phạt 1000 lần , việc sử phạt lúc nào cũng phải mạnh tay hơn khen thưởng

Tốc độ học xác định mức độ mà thông tin mới ghi đè lên thông tin cũ. Tốc độ học sau đây được chọn : do mình quy định

Hệ số chiết khấu (discount factor) xác định tầm quan trọng của tương lai phần thưởng. Giá trị này được đưa ra dưới đây.

## Hình ảnh 1 lần chạy tiêu chuẩn

Nếu chạm ống



Nếu không chạm ống



# Yêu cầu cấu hình chạy phần mềm

Yêu cầu cấu hình tối thiểu:

* CPU từ 1.6 GHz trở lên
* RAM từ 4 GB

Có Microsoft .NET Framework 4.5.2

**Phần mềm**

Visual Studio Code rất nhẹ (54 MB với bản dành cho Windows)

**Cài đặt Python:**

**Tải Python:**

Truy cập trang chủ Python và tải phiên bản phù hợp với hệ điều hành của bạn (Windows, macOS, hoặc Linux).

Cài đặt Python:

Mở tệp tải về Python và làm theo hướng dẫn để cài đặt Python interpreter trên máy tính của bạn. Trong quá trình cài đặt, đảm bảo bạn chọn tùy chọn "Add Python to PATH" để có thể truy cập Python từ Command Prompt hoặc Terminal.

# Tài liệu tham khảo

1. [FlappyQ.pdf (bilkent.edu.tr)](https://kilyos.ee.bilkent.edu.tr/~eee546/FlappyQ.pdf)(Flappy BirdQLearning)

1.([kilyos.ee.bilkent.edu.tr/~eee546/FlappyQ.pdf](https://kilyos.ee.bilkent.edu.tr/~eee546/FlappyQ.pdf))(Lần truy cập cuối cùng 12h ngày 30/11/2023)

1. <https://www.javatpoint.com/reinforcement-learning#Q-Learning> ([Reinforcement Learning Tutorial - Javatpoint](https://www.javatpoint.com/reinforcement-learning" \l "Q-Learning))(Lần truy cập cuối cùng 12h ngày 30/11/2023)
2. https://viblo.asia/p/reinforcement-learning-q-learning-63vKjO7VZ2R(Reinforcement Learning: Q-Learning)(Lần truy cập cuối cùng 12h ngày 30/11/2023)
3. <https://www.youtube.com/watch?v=yyNOa9aB77E&list=LL&index=3&t=381s> ([Understanding Reinforcement Learning with Flappy Bird - YouTube](https://www.youtube.com/watch?v=yyNOa9aB77E&list=LL&index=3&t=381s))(Lần truy cập cuối cùng 12h ngày 30/11/2023)
4. https://www.youtube.com/watch?v=rcgJpEfFYik(Lần truy cập cuối cùng 12h ngày 30/11/2023)