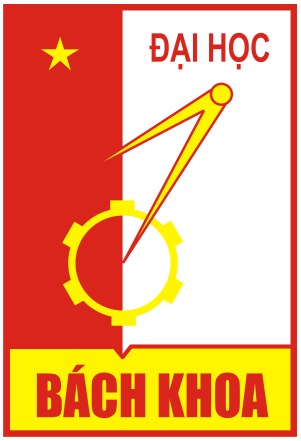
TRƯỜNG ĐẠI HỌC BÁCH KHOA HÀ NỘI
VIỆN TOÁN ỨNG DỤNG VÀ TIN HỌC



BÁO CÁO BÀI TẬP LỚN
CÁC MÔ HÌNH NGẪU NHIÊN VÀ ỨNG DỤNG

ĐỀ TÀI: REINFORCEMENT LEARNING - MDP VÀ BÀI TOÁN FROZENLAKE

Giáo viên: TS. Nguyễn Thị Ngọc Anh

Sinh viên:
PHÙNG ANH HÙNG - 20173150
NGUYỄN ĐỨC VƯỢNG - 20173603
NGÔ VIỆT TRUNG - 20173415

Hà Nội - Tháng 7/2020

Lời nói đầu

Học tăng cường (Reinforcement Learning- RL) bao gồm các loại kĩ thuật để tìm ra chính sách nhằm tối đa hóa phần thưởng nhận được. Nó giúp ta tìm ra được chính sách hợp lý cho từng trạng thái. Trong bài báo cáo này, nhóm chúng em xin phép trình bày cách áp dụng mô hình quyết định Markov (Markov Decision Process) trong việc tạo ra hai thuật toán Lặp chính sách và Lặp giá trị. Từ đó sử dụng hai thuật toán để giải bài toán Hồ băng (Frozen Lake).

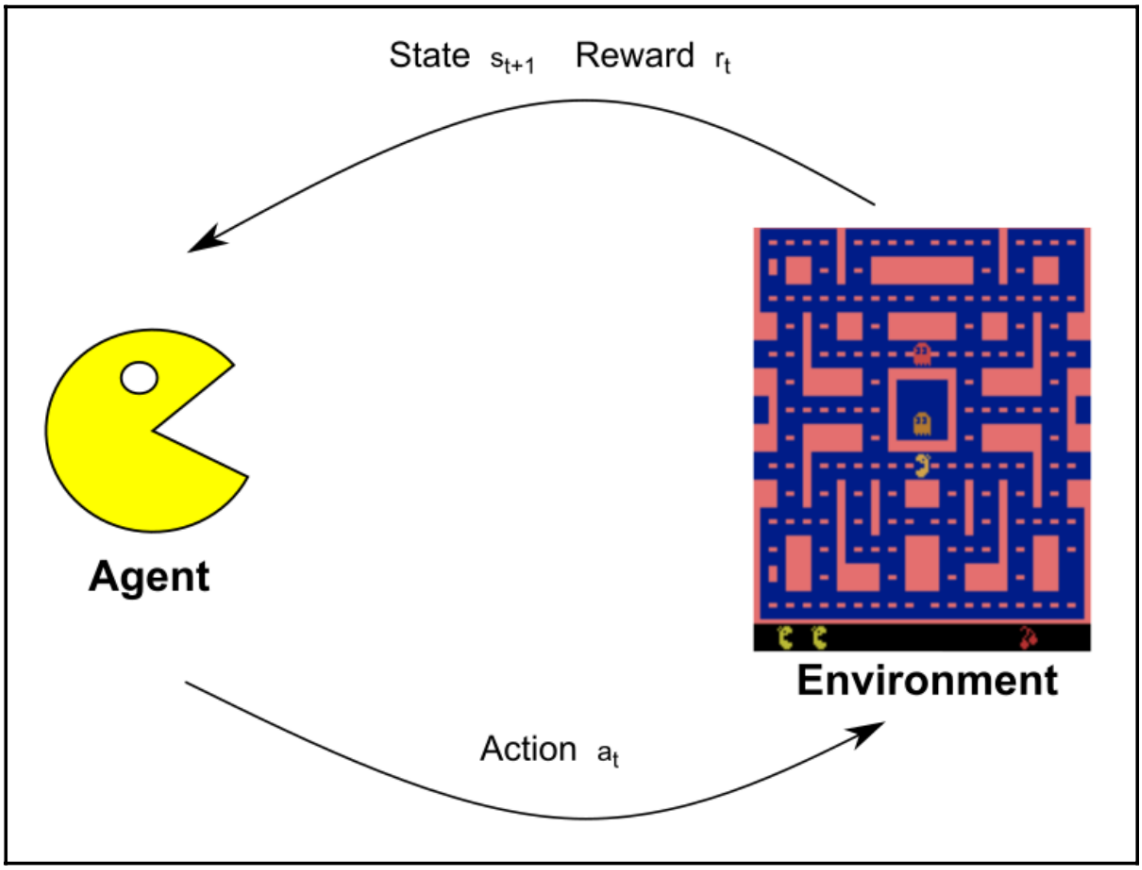
Báo cáo có thể còn nhiều thiếu sót, chúng em hi vọng có thể nhận được sự nhận xét và đánh giá từ thầy cô để có thể rút kinh nghiệm và hoàn thiện hơn. Chúng em xin cảm ơn.

Mục lục

1 Giới thiệu về Reinforcement Learning và ứng dụng

1.1 Reinforcement Learning là gì?

Reinforcement Learning là một lĩnh vực học máy liên quan đến việc ra quyết định tuần tự, nhằm đạt được mục tiêu mong muốn. Một vấn đề Reinforcement Learning được cấu thành bởi một người ra quyết định gọi là một máy (Agent) và thế giới vật lý hoặc ảo mà máy tương tác, được gọi là môi trường (Environment). Máy tương tác với môi trường dưới dạng hành động (Action) dẫn đến hiệu ứng. Sau đó, môi trường sẽ phản hồi cho máy một trạng thái (State) mới và phần thưởng (Reward). Hai tín hiệu này là kết quả của hành động được thực hiện bởi máy. Cụ thể, phần thưởng là một giá trị cho biết hành động đó tốt hay xấu và trạng thái là đại diện hiện tại của máy và môi trường. Chu trình này được hiển thị trong sơ đồ sau:



Trong sơ đồ này, máy được đại diện bởi PacMan, dựa trên trạng thái hiện tại của môi trường, chọn hành động nào cần thực hiện. Hành vi của nó sẽ ảnh hưởng đến môi trường, như vị trí của nó và của kẻ thù, sẽ được môi trường trả lại dưới dạng một trạng thái mới và phần thưởng. Chu kỳ này được lặp lại cho đến khi trò chơi kết thúc.
Mục tiêu cuối cùng của máy là tối đa hóa tổng số phần thưởng tích lũy được trong thời gian vòng đời của nó. Hãy đơn giản hóa ký hiệu: nếu a_t là hành động tại thời điểm t

và r_t là phần thưởng tại thời điểm t, máy sẽ thực hiện các hành động $a_0, a_1, \ldots, a_t$ để tối đa hóa tổng của tất cả các phần thưởng $\sum_{i=0}^{t} r_i$
Để tối đa hóa phần thưởng tích lũy, máy phải học hành vi tốt nhất trong mọi tình huống. Để làm như vậy, máy phải tối ưu hóa cho một loạt các hành động dài hạn trong khi thực hiện từng hành động. Trong môi trường có nhiều trạng thái và hành động rời rạc hoặc liên tục, việc học là rất khó khăn vì máy phải chịu trách nhiệm cho từng tình huống. Để làm cho vấn đề trở nên phức tạp hơn, phần thưởng có thể rất thưa thớt và bị trì hoãn, làm cho quá trình học tập trở nên khó khăn hơn.

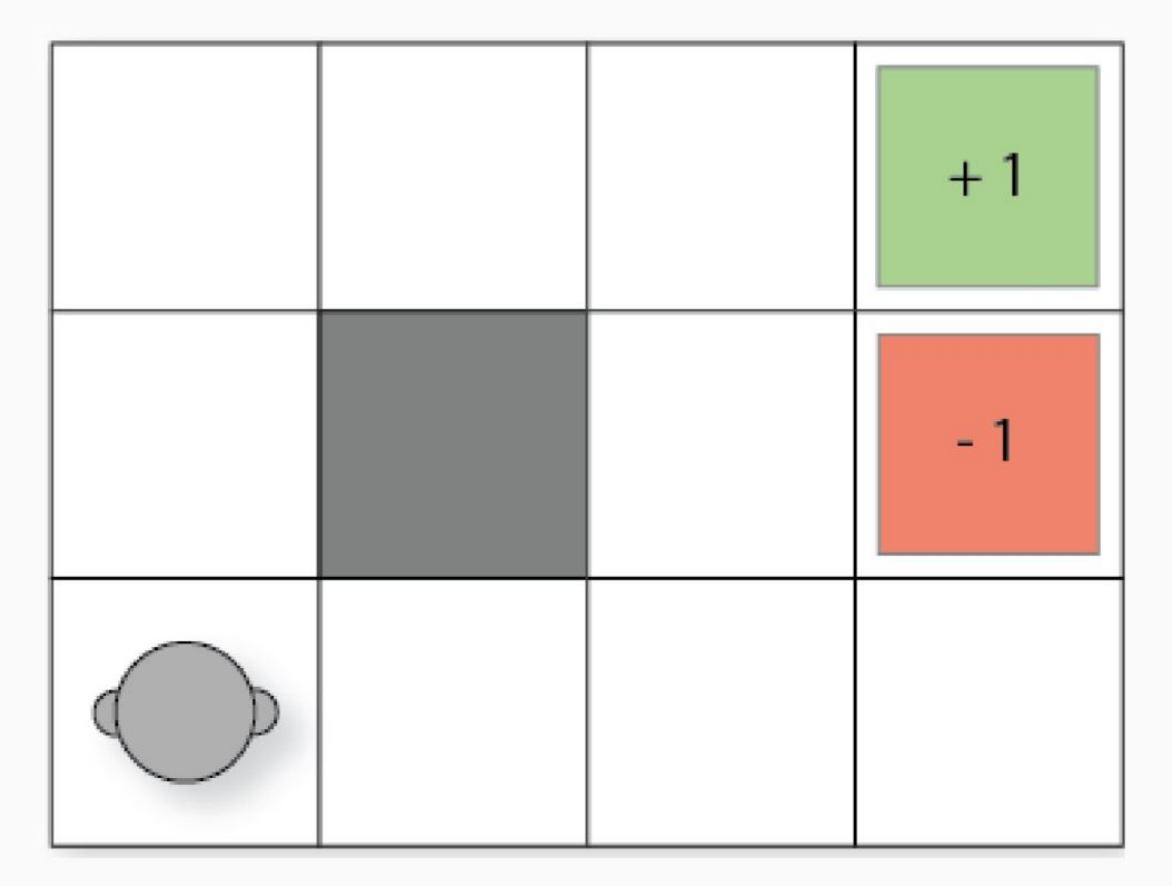
1.2 Các thành phần của Reinforcement Learning

Như chúng ta đã biết, một máy tương tác với môi trường của nó bằng các hành động. Điều này sẽ khiến môi trường thay đổi và phản hồi cho máy một phần thưởng tỷ lệ thuận với chất lượng của các hành động và trạng thái mới của máy. Thông qua thử nghiệm và sai sót, máy dần dần học được hành động tốt nhất để thực hiện trong mọi tình huống để về lâu dài, nó sẽ đạt được phần thưởng tích lũy lớn hơn. Trong khung RL, việc lựa chọn hành động ở một trạng thái cụ thể được thực hiện bởi một chính sách (Policy) và phần thưởng tích lũy có thể đạt được từ trạng thái đó được gọi là hàm giá trị (Value Function). Tóm lại, nếu một máy muốn hành xử tối ưu, thì trong mọi tình huống, chính sách phải chọn hành động sẽ đưa nó đến trạng thái tiếp theo với giá trị cao nhất.

Các thành phần của Reinforcement Learning:

- Agent: máy - người ra quyết định
- Environment: thế giới vật lý hoặc ảo mà máy tương tác
- Action: hành động mà máy tương tác với môi trường
- State: trạng thái của máy trong môi trường
- Reward: sau mỗi lần di chuyển của máy, môi trường sẽ gửi lại một thông số cho biết hành động đó tốt như thế nào đối với tác nhân, đó gọi là phần thưởng
- Policy: Chính sách xác định cách tác nhân chọn một hành động tiếp theo dựa vào trạng thái hiện tại. Chính sách chọn hành động mà tối đa hóa phần thưởng tích lũy từ trạng thái đó, không phải với phần thưởng lớn hơn ngay lập tức. Nó quan tâm đến việc tìm kiếm mục tiêu dài hạn của các máy.
- Value function: Hàm giá trị đại diện cho chất lượng lâu dài của một trạng thái. Đây là phần thưởng tích lũy được dự kiến trong tương lai nếu máy bắt đầu từ một trạng thái nhất định.

1.3 Ví dụ về Reinforcement Learning



Nhiệm vụ của con robot (Agent) là đi đến ô đích màu xanh, tránh ô phạt màu đỏ và ô xám là chướng ngại vật không được phép đi vào. Robot tương tác với môi trường bằng các hành động trái/phải/lên/xuống. Sau mỗi hành động, môi trường trả lại cho robot một trạng thái (ở ví dụ này là vị trí của robot) và phần thưởng tương ứng với trạng thái đó (+1 nếu đi vào xanh, -1 nếu đi vào ô đỏ và 0 nếu ở ô trắng). Khi robot đến ô xanh hoặc đỏ trò chơi kết thúc; một loạt các tương tác giữa robot và môi trường từ thời điểm bắt đầu đến lúc này được gọi là một màn chơi (Episode). Trong một màn chơi, máy (Agent) sẽ cố gắng chọn ra các hành động tối ưu để tối đa hóa phần thưởng nhận được sau mỗi màn chơi. Cách mà máy chọn những hành động đó là chính sách (Policy), ví dụ: "đi ngẫu nhiên", "đi men theo rìa" hoặc "hướng về ô đích". Mục đích của Reinforcement Learning là tìm ra chính sách tốt nhất cho con robot.

1.4 Ứng dụng của Reinforcement Learning

Một số ứng dụng của Reinforcement Learning:

- Rô-bốt trong công nghiệp tự động hóa và hàng không.
- Xây dựng chiến lược kinh doanh.

- Máy học và xử lí dữ liệu.
- Tạo hệ thống đào tạo cung cấp hướng dẫn và tài liệu theo yêu cầu của sinh viên.
- Q-learning và Deep Q-learning.

Trong thực tế gần đây, Reinforcement Learning đã đạt được những thành tựu đáng kể sau khi Google DeepMind phát triển các thuật toán dựa trên RL và có thể thắng được các tuyển thủ vô địch thế giới trong một số trò chơi thể thao điện tử, điển hình như:

- Thuật toán AlphaGo trong môn cờ vây.
- Phần mềm AlphaStar trong môn Starcraft.
- Phần mềm Open AI FIVE trong môn Dota 2.

2 Quá trình quyết định Markov

2.1 Giới thiệu MDP

- Trong mô hình MDP, một môi trường được mô hình hóa như một tập hợp các trạng thái và hành động. Các hành động có thể được thực hiện để kiểm soát trạng thái hệ thống. Mục tiêu là kiểm soát hệ thống theo cách mà một số tiêu chí tối ưu nào đó. Nhiều vấn đề như lập kế hoạch, điều khiển robot và giải mội trò chơi đã được mô hình hóa thành công theo thuật ngữ MDP. MDP đã trở thành hình thức tiêu chuẩn thực tế cho việc học cách thực hiện tuần tự.
- Để mô tả rõ hơn về MDP, em xin lấy một ví dụ về bài toán Robot dọn rác. Một robot dọn rác có hai trạng thái của pin là "mạnh" và "yếu". Tại trạng thái pin "yếu" robot có thể thực hiện một trong ba hành động "tìm" (rác), "đợi" (rác), "sạc". Tại trạng thái pin "mạnh" thì robot chỉ có thể thực hiện hành động "tìm" hoặc "đợi". Trong bài toán này, khi ta thực hiện hành động "tìm" ở trạng thái pin "yếu", có khả năng nhất định là pin robot sẽ cạn kiệt và ngừng hẳn hoạt động, khi đó cần bỏ thêm nhiều thời gian để sạc hoặc thay pin và khởi động lại robot. Chuỗi các hành động của robot một chuỗi Markov, và mục tiêu của ta là tìm ra một nguyên tắc nào đó cho các hành động sao cho robot đạt hiệu suất làm việc tốt nhất.

2.2 Định nghĩa MDP

MDP thể hiện việc đưa ra các quyết định theo thứ tự, trong đó các hành động ảnh hưởng đến trạng thái và kết quả.
Một MDP được biểu diễn bằng một bộ 4 dữ liệu (S, A, P, R):

- S là không gian trạng thái với một tập hữu hạn các trạng thái.
- A là một tập hữu hạn các hành động.
- P là hàm chuyển tiếp, xác định xác suất đạt trạng thái thái s' từ trạng thái s thông qua hành động a. Kí hiệu:

  $$P(s', s, a) = p(s'|s, a)$$

  , hàm chuyển đổi bằng với xác suất xảy ra trạng thái s với điều kiện đã cho là trạng thái s và hành động a.
- R là hàm phần thưởng, xác định giá trị nhận được khi chuyển sang trạng thái s' từ trạng thái s thông qua hành động a.

Sử dụng ví dụ trên hình 2.1, ta có tập trạng thái $S = \{$ *"yếu"*, *"mạnh"*$\}$ và tập các hành động $A = \{$ *"sạc"*, *"tìm"*, *"đợi"*$\}$. Mũi tên có hướng mô tả sự chuyển đổi giữa hai trạng thái thông qua một hành động với xác suất nào đó. Ví dụ như xác suất để chuyển từ trạng thái *"mạnh"* sang trạng thái *"yếu"* thông qua hành động *"tìm"* là $1 - a$.
Từ ví dụ trên, ta có thể thấy rõ rằng MDP được tạo từ một chuỗi các trạng thái,

hành động $(S_0, A_0, S_1, A_1, \dots)$, nơi mà các trạng thái tuân theo hàm chuyển trạng thái $p(s'|s, a)$. Trạng thái trong tương lai chỉ phụ thuộc vào trạng thái hiện tại chứ không hề phụ thuộc vào các trạng thái trước đó.
Mục tiêu của MDP là tìm ra một chính sách π có thể tối đa hóa phần thưởng tích lũy $\sum_{t=0}^{\infty} R_\pi(s_t, s_{t+1})$, với $R_\pi(s_t, s_{t+1})$ là phần thưởng nhận được khi từ trạng thái s_t sang s_{t+1} khi tuân theo chính sách π.
Chính sách chọn hành động có thể thực hiện trong một tình huống nhất định. Ta có thể phân loại chính sách thành chính sách xác định và chính sách ngẫu nhiên.
Chính sách xác định kí hiệu là $a_t = \mu(s_t)$, chính sách ngẫu nhiên kí hiệu là $a_t \sim \pi(.|s_t)$.

2.3 Hàm trả về

Khi chạy một chính sách trong một MDP, ta có chuỗi trạng thái và hành động $(S_0, A_0, S_1, A_1, \dots)$ gọi là quỹ đạo hoặc khai triển kí hiệu là τ. Trong mỗi quỹ đạo, một chuỗi các phần thưởng sẽ thu được do kết quả của các hành động. Hàm trả về là hàm biểu thị dạng đơn giản của các phần thưởng:

$$G(\tau) = r_0 + r_1 + r_2 + ... = \sum_{t=0}^{\infty} r_t \tag{1}$$

Với cách biểu diễn $G(\tau)$ như trên, kết quả trả về từ các quỹ đạo có khả năng sẽ đều là vô cùng. Điều này đồng nghĩa với việc hàm trả về không mang lại một thông tin nào. Khi đó ta cần một cách biểu diễn khác của hàm trả về. Giải pháp ở đây là ta tăng giá trị của phần thưởng ngắn hạn và giảm tầm quan trọng của các phần thưởng trong tương lai xa. Ta thực hiện điều này bằng cách sử dụng hệ số chiết khấu γ có giá trị nằm trong khoảng từ 0 đến 1. Khi đó $G(\tau)$ được điều chỉnh lại như sau:

$$G(\tau) = r_0 + \lambda r_1 + \lambda^2 r_2 + ... = \sum_{t=0}^{\infty} \lambda^t r_t \tag{2}$$

Trong trường hợp xét trên số bước hữu hạn, $G(\tau)$ có dạng:

$$G(\tau) = r_0 + \lambda r_1 + \lambda^2 r_2 + ... = \sum_{t=0}^{k} \lambda^t r_t \tag{3}$$

Kí hiệu $G_t = G_t(\tau)$ là tổng trả về tính từ bước thứ t. Ta có công thức:

$$G_t(\tau) = r_t + \lambda r_{t+1} + \lambda^2 r_{t+2} + ... \tag{4}$$

Từ đó ta suy ra công thức sau:

$$G_t(\tau) = r_t + \lambda G_{t+1}(\tau) \tag{5}$$

Hay viết đơn giản là

$$G_t = r_t + \lambda G_{t+1} \tag{6}$$

2.4 Hàm giá trị

Hàm trả về $G(\tau)$ cho ta cái nhìn về phần thưởng nhận được qua cả quỹ đạo, nhưng nó không cho ta biết được chất lượng của một trạng thái nhất định. Chỉ số chất lượng này rất quan trọng vì giá trị của trạng thái có thể được chính sách sử dụng để chọn hành động tốt nhất tiếp theo. Hàm giá trị có trách nhiệm tính giá trị kì vọng vào một trạng thái theo một chính sách nào đó. Hàm giá trị được định nghĩa như sau:

$$V_\pi(s) = E_\pi[G|s_0 = s] = E_\pi[\sum_{t=0}^{k} \lambda^t r_t | s_0 = s] \tag{7}$$

Tương tự ta có hàm giá trị hành động, cũng là kì vọng phần thưởng từ một trạng thái nhưng bao gồm thêm hành động thực hiện từ trạng thái đó. Hàm giá trị hành động được định nghĩa như sau:

$$Q_\pi(s, a) = E_\pi[G|s_0 = s, a_0 = a] = E_\pi[\sum_{t=0}^{k} \lambda^t r_t | s_0 = s, a_0 = a] \tag{8}$$

2.5 Phương trình Bellman

Ta có thể tính V và Q bằng cách chạy các quỹ đạo tuân theo chính sách π và lấy trung bình các giá trị thu được. Kĩ thuật này có hiệu quả tuy nhiên lại rất tốn kém vì nó đòi hỏi giá trị phần thưởng từ cả quỹ đạo. Phương trình Bellman thực hiện phép tính Q và V bằng cách sử dụng phần thưởng thu được ở trạng thái hiện tại và trạng thái tiếp theo. Phương trình Bellman áp dụng công thức đệ quy (2.6) như sau:

$$\begin{aligned} V_\pi(s) &= E_\pi[G|s_0 = s] \\ &= E_\pi[r_t + \gamma G_{t+1}|s_0 = s] \\ &= E_\pi[r_t + \gamma V_\pi(s_{t+1}|s_t = s, a_t \sim \pi(s_t))] \end{aligned} \tag{9}$$

$$\begin{aligned} Q_\pi(s, a) &= E_\pi[G_t|s_0 = s, a_t = a] \\ &= E_\pi[r_t + \gamma G_{t+1}|s_t = s, a_t = a] \\ &= E_\pi[r_1 + \gamma Q_{pi}(s_{t+1}, a_{t+1})|s_t = s, a_t = a] \end{aligned} \tag{10}$$

2.6 Đánh giá và cải thiện chính sách

Để tìm chính sách tối ưu, trước tiên ta cần tìm hàm giá trị tối ưu. Một thủ tục lặp dùng để tìm hàm giá trị tối ưu, nó tạo ra một chuỗi lặp $V_0, \ldots, V_k$ nhằm cải thiện giá trị cho chính sách π. Bằng cách áp dụng phương trình Bellman, ta có thể cải thiện giá trị có thể nhận được từ một trạng thái theo công thức:

$$V_{k+1}(s) = E_\pi[r_t + \gamma V_k(s_{t+1})|s_t = s] = \sum_a \pi(s, a) \sum_{s', r} p(s'|s, a)[r + \gamma V_k(s')] \tag{11}$$

Hàm giá trị V chỉ có thể được cập nhật khi ta đã biết hết các hàm chuyển trạng thái

và phần thưởng của mọi trạng thái lẫn hành động.
Khi các hàm giá trị được cải thiện, ta có thể sử dụng nó để tìm được chính sách tốt hơn. Quy trình này được gọi là cải thiện chính sách, ta tìm được π' theo công thức:

$$\pi' = argmax_a Q_\pi(s,a) = argmax_a \sum_{s',r} p(s'|s,a)[r + \gamma V_\pi(s')] \tag{12}$$

Công thức trên tạo ra chính sách π' từ hàm giá trị V_π của chính sách ban đầu π. Chính sách π' luôn tốt hơn chính sách π, và chính sách sẽ tối ưu khi và chỉ khi V cũng tối ưu. Sự kết hợp giữa đánh giá chính sách và cải thiện chính sách tạo ra hai thuật toán để tính toán chính sách tối ưu là thuật toán lặp chính sách và thuật toán lặp giá trị, hai thuật toán sẽ được nêu cụ thể ở phần sau.

2.7 Thuật toán lặp chính sách

Thuật toán lặp chính sách thực hiện lặp giữa các lần đánh giá chính sách, cập nhật V_π theo chính sách π hiện tại. Đồng thời thuật toán sử dụng công thức (11) và (12) để tính toán chính sách π' bằng hàm giá trị V_π đã được cải thiện. Cuối cùng, sau n chu kì, thuật toán sẽ dẫn đến một chính sách tối ưu π^*.
Nội dung thuật toán:

Khởi tạo $V_\pi(s)$ và $\pi(s)$ cho mỗi trạng thái s
while π không ổn định **do**
    while V_π không ổn định **do** ▷ Đánh giá chính sách
        for mỗi trạng thái s **do**

$$V_\pi(s) = \sum_{s',r} p(s'|s,\pi(a))[r + \gamma V_\pi(s')]$$

        end for
    end while
    for mỗi trạng thái s **do** ▷ Cải thiện chính sách

$$\pi = argmax_a \sum_{s',r} p(s'|s,a)[r + \gamma V_\pi(s')]$$

    end for
end while

2.8 Thuật toán lặp giá trị

Khác với thuật toán lặp chính sách, thuật toán lặp giá trị không tính kì vọng của hàm giá trị V_π rồi sử dụng chúng để tìm chính sách π' tốt hơn. Thay vào đó, thuật toán kết hợp hai công thức đánh giá chính sách và cải thiện chính sách trong một công thức. Cụ thể, thuật toán sẽ cập nhật giá trị của trạng thái bằng cách chọn hành động tốt nhất có thể ở trạng thái đó theo công thức sau:

$$V_{k+1}(s) = max_a \sum_{s',r} p(s'|s,a)[r + \gamma V_k(s')] \tag{13}$$

Nội dung thuật toán:

1: Khởi tạo $V(s)$ cho mỗi trạng thái s
2: **while** V chưa ổn định **do** ▷ Lặp giá trị
3: **for** mỗi trạng thái s **do**
4:

$$V(s) = max_a \sum_{s',r} p(s'|s,a)[r + \gamma V(s')]$$

5: **end for**
6:

$$\pi = argmax_a \sum_{s',r} p(s'|s,a)[r + \gamma V(s)]$$

7: **end while**

Chính sách tối ưu sẽ có kết quả như sau:

$$\pi^* = argmax_a \sum_{s',r} p(s'|s,a)[r + \gamma V^*(s)] \tag{14}$$

3 Ứng dụng thuật toán giải bài toán Frozen Lake

3.1 Nội dung bài toán

Bài toán Frozen Lake (Hồ băng): Ta có một mặt hồ bị đóng băng được chia thành các ô vuông nhỏ. Người chơi có nhiệm vụ đi từ ô khởi đầu đến ô kết thúc của hồ. Trên mặt hồ có một vài ô là hố, nếu người chơi đi vào hố, người chơi sẽ thua. Đặc biệt, người chơi chỉ có một xác suất nhất định là có thể đi theo hướng mình muốn. Trong bài toán Frozen Lake của báo cáo, em xin đưa ra những quy ước sau:

- Các hướng trong bài toán:
  - 0 là sang trái
  - 1 là đi xuống
  - 2 là sang phải
  - 3 là đi lên
- Khi người chơi chọn một hướng để đi, xác suất bước được chia đề cho ba hướng là hướng đã chọn và hai hướng hai bên.
- Hồ băng sử dụng trong bài toán có kích thước $5 * 5$, được mô tả bằng ma trận như sau:

$$\begin{bmatrix} S & F & F & F & H \\ F & H & F & F & F \\ F & F & H & F & H \\ F & F & F & F & F \\ F & H & F & F & G \end{bmatrix}$$

  Trong đó:
  - S là ô khởi đầu
  - G là ô đích, người chơi thắng khi tới ô này
  - F là ô an toàn, có thể đi qua
  - H là ô hố, người chơi sẽ thua nếu đi vào ô này
- Phần thưởng cho việc đi vào ô đích là +1 và các ô còn lại là 0.

3.2 Áp dụng thuật toán lặp chính sách

Áp dụng thuật toán lặp chính sách, ta thu được kết quả như sau:
Sau 7 lần lặp chính sách. Ta thu được chính sách giúp thắng 579 trên 1000 lần chơi

Hàm giá trị:

$$\begin{bmatrix} 0.36622453 & 0.32930569 & 0.30253772 & 0.28507986 & 0. \\ 0.37759253 & 0. & 0.19102399 & 0.27636208 & 0.13610881 \\ 0.40054318 & 0.284604390. & 0.36137257 & 0. &] \\ 0.43572637 & 0.46192058 & 0.67944319 & 0.8187138 & 0.90075766 \\ 0.42283587 & 0. & 0.77830134 & 0.90075766 & 0. \end{bmatrix}$$

Chính sách tối ưu:

$$\begin{bmatrix} 0 & 3 & 3 & 0 & 0 \\ 0 & 0 & 2 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 \\ 1 & 3 & 1 & 1 & 1 \\ 0 & 0 & 2 & 2 & 0 \end{bmatrix}$$

3.3 Áp dụng thuật toán lặp giá trị

Áp dụng thuật toán lặp chính sách, ta thu được kết quả như sau:
Sau 106 lần lặp giá trị. Ta thu được chính sách giúp thắng 589 trên 1000 lần chơi
Hàm giá trị:

$$\begin{bmatrix} 0.36621021 & 0.32929648 & 0.30253222 & 0.2850765 & 0. \\ 0.37757871 & 0. & 0.19102151 & 0.27635968 & 0.13610773 \\ 0.40053051 & 0.28459772 & 0. & 0.36137105 & 0. \\ 0.43571592 & 0.46191353 & 0.67943923 & 0.81871175 & 0.90075661 \\ 0.42282392 & 0. & 0.77829875 & 0.90075661 & 0. \end{bmatrix}$$

Chính sách tối ưu:

$$\begin{bmatrix} 0 & 3 & 3 & 0 & 0 \\ 0 & 0 & 2 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 \\ 1 & 3 & 1 & 1 & 1 \\ 0 & 0 & 2 & 2 & 0 \end{bmatrix}$$

Như vậy, ta thấy cả hai thuật toán Lặp chính sách và thuật toán Lặp giá trị đều hội tụ về cùng một kết quả. Bằng cách sử dụng thuật toán, ta tìm được chính sách tối ưu nhất để có thể thắng trò chơi Frozen Lake.

Danh mục tài liệu tham khảo

Tài liệu

[1] Andrea Lonza, *Reinforcement Learning Algorithms with Python (2019)*.

[2] Taweh Beysolow, *Applied Reinforcement Learning with Python. With OpenAI Gym, Tensorflow and Keras-Apress (2019)*.

[3] Martijn van Otterlo and Marco Wiering, *Reinforcement Learning and Markov Decision Processes*.