MỤC LỤC

MỤC LỤC i

DANH MỤC HÌNH ẢNH iii

DANH MỤC BẢNG iv

Chương 1 Giới thiệu 1

Chương 2 Kiến thức nền tảng 5

2.1 Các thành phần cơ bản của học tăng cường 5

2.1.1 Agent và môi trường 5

2.1.2 Returns 7

2.2 Mô hình Markov Decision Processes 8

2.2.1 Định nghĩa mô hình Markov Decision Processes 8

2.2.2 Chính sách và hàm giá trị 11

2.2.3 Hàm giá trị tối ưu 14

2.3 Quy trình lặp chính sách 16

2.3.1 Phương pháp đánh giá chính sách 17

2.3.2 Phương pháp cải thiện chính sách 21

Chương 3 Kết hợp học sâu với học tăng cường 23

Chương 4 Kết quả thực nghiệm 24

4.1 Giới thiệu Arcade Learning Environment 24

4.2 Giới thiệu cấu trúc mạng và các siêu tham số đã chọn 24

4.3 Kết quả thực nghiệm 24

Chương 5 Kết luận và hướng phát triển **25**

TÀI LIỆU THAM KHẢO **26**

DANH MỤC HÌNH ẢNH

1.1 Hình ảnh các game trên hệ máy Atari 3

2.1 Quá trình tương tác giữa hệ thông và môi trường 6
2.2 Đồ thị minh họa chuyển trạng thái cho robot thu gom 10
2.3 Đồ thị minh họa quan hệ giữa những hàm giá trị 12
2.4 Đồ thị minh họa cho hàm giá trị 12
2.5 Đồ thị minh họa quan hệ giữa những hàm giá trị tối ưu 15
2.6 Đồ thị minh họa phương trình Bellman trong hàm giá trị tối ưu 15
2.7 Quy trình lặp chính sách 17
2.8 Cập nhật hàm giá trị bằng quy hoạch động 18
2.9 Cập nhật hàm giá trị bằng phương pháp Monte Carlo 21

DANH MỤC BẢNG

Chương 1

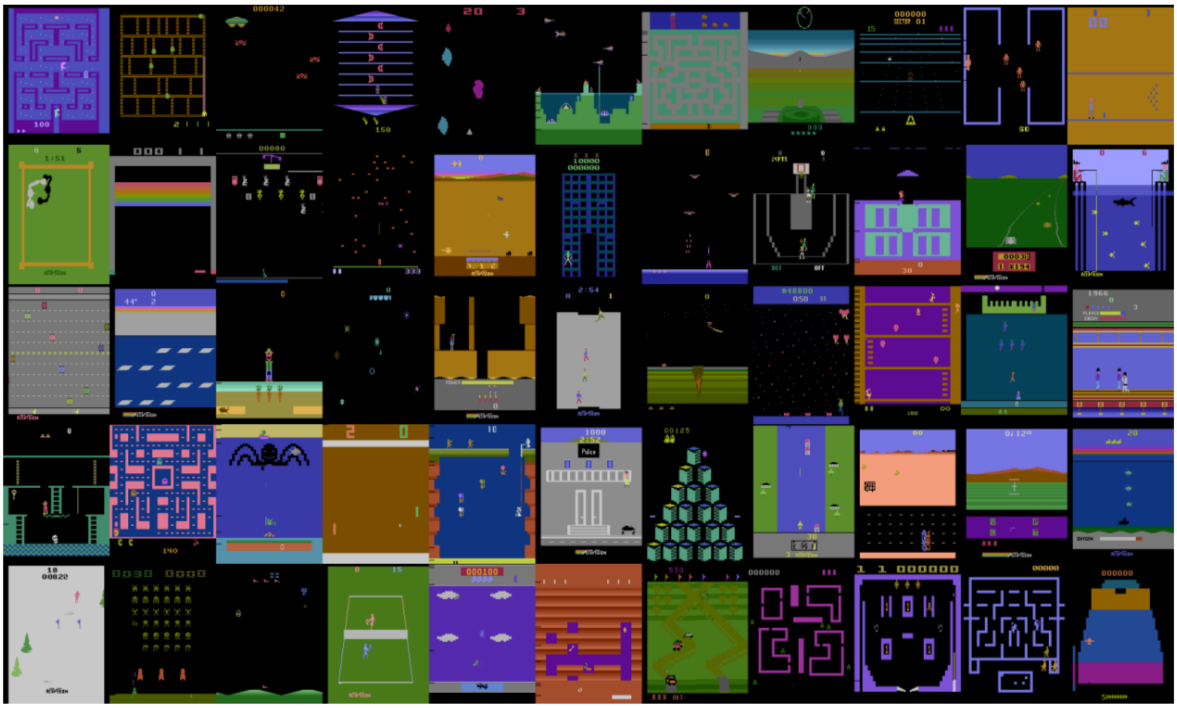
Giới thiệu

Những năm gần đây, **học tăng cường** (Reinforcement learning) liên tục đạt được những thành tựu quan trọng trong lĩnh vực Trí tuệ nhân tạo (Artificial Intelligence). Những đóng góp nổi bật của phương pháp này bao gồm: tự động điều khiển robot di chuyển, điều khiển mô hình máy bay trực thăng, hệ thống chơi cờ vây... Trong số các thành tựu này, hệ thống chơi cờ vây với khả năng chiến thắng những kỳ thủ hàng đầu thế giới là một cột mốc quan trọng của lĩnh vực Trí tuệ nhân tạo. Dù vậy, học tăng cường không phải là một phương pháp mới được phát triển gần đây. Nền tảng lý thuyết của học tăng cường đã được xây dựng từ những năm 1980.

Được xây dựng nhằm mô phỏng quá trình học của con người, ý tưởng chính của học tăng cường là tìm cách lựa chọn hành động *tối ưu* để nhận được **nhiều nhất giá trị điểm thưởng** (Reward). Giá trị điểm thưởng này có ý nghĩa tương tự cảm nhận của con người về môi trường. Khi một đứa trẻ bắt đầu "học" về thế giới xung quanh của mình, những cảm giác như đau đớn (ứng với điểm thưởng thấp) hay vui sướng (điểm thưởng cao) chính là mục tiêu cần tối ưu của việc học. Một điểm quan trọng của học tăng cường là nó được xây dựng với ít giả định nhất có thể về môi trường xung quanh. Hệ thống sử dụng học tăng cường (Agent) không cần biết cách thức hoạt động của môi trường để hoạt động. Ví dụ như để điều khiển robot tìm được đi trong mê cung, hệ thống không cần biết mê cung được xây dựng thế nào hay kích thước là bao nhiêu. Việc hạn chế tối đa những ràng buộc về dữ liệu đầu vào của bài toán học tăng cường giúp cho phương pháp này có thể áp dụng vào nhiều bài toán thực tế.

Học tăng cường được xem là một nhánh trong lĩnh vực máy học ngoài hai nhánh: học có giám sát và học không có giám sát. Trong bài toán học có giám sát, dữ liệu thường được gán nhán thủ công sẵn và việc chủ yếu của hệ thống là làm sao dự đoán chính xác các nhãn đó với dữ liệu mới. Các nhãn này có thể xem như là sự hướng dẫn trong quá trình học; tính đúng sai của việc học lúc này có thể được xác định dựa vào kết quả dự đoán của hệ thống và nhãn đúng của dữ liệu. Tiếp theo đối với những bài toán học không có giám sát, dữ liệu học thường không được gán nhãn nên công việc của việc học là phải tự tìm ra được cấu trúc "ẩn" bên dưới dữ liệu đó. Khác với hai loại bài toán vừa nêu, trong bài toán học tăng cường, hệ thống *không nhận được nhãn thực sự* (tức hành động tối ưu của tình huống hiện tại) mà chỉ nhận được điểm thưởng từ môi trường. Điểm thưởng lúc này chỉ thể hiện mức độ "tốt/xấu" của hành động vừa chọn chứ không nói lên hành động đó có phải là hành động tối ưu hay không. Điểm thưởng này thông thường rất thưa: ta có thể chỉ nhận được điểm thưởng có ý nghĩa (khác không) sau hàng nghìn hành động. Ngoài ra, giá trị điểm thưởng thường là không đơn định và rất nhiễu: cùng một hành động tại cùng một trạng thái, ta có thể nhận được điểm thưởng khác nhau vào hai thời điểm khác nhau. Đây cũng chính là những khó khăn cơ bản của bài toán học tăng cường.

Các trò chơi điện tử thường hay có điểm số mà người chơi cần phải tối ưu hoá. Đặc điểm này trùng với yêu cầu của bài toán học tăng cường, vì vậy các trò chơi này cũng chính là những ứng dụng tự nhiên nhất của phương pháp học tăng cường. Trong luận văn này, chúng em áp dụng phương pháp học tăng cường nhằm xây dựng **hệ thống tự động chơi các game** trên hệ máy Atari. Dữ liệu đầu vào của hệ thống chỉ bao gồm các frame ảnh RGB cùng với điểm số hiện tại. Từ hình ảnh thô này, hệ thống cần tìm cách chơi sao cho điểm số cuối màn chơi (Episode) là lớn nhất có thể. Hệ thống hoàn toàn không biết quy luật của game trước khi bắt đầu quá trình học mà phải tự tìm hiểu quy luật và chiến thuật chơi tối ưu. Lý do luận văn sử dụng game của máy Atari là vì các game này có quy luật chơi tương đối đơn giản nhưng lại rất đa dạng. Mỗi màn chơi thường có độ dài vừa phải (từ 2 - 15 phút) và số hành động có ý nghĩa không quá nhiều (18 hành động). Ngoài ra, các trò chơi này có thể được giả lập trên máy vi tính với tốc độ cao, giúp quá trình học được tăng tốc.



Hình 1.1: Hình ảnh các game trên hệ máy Atari

Một số khó khăn trước mắt có thể thấy ở bài toán tự động chơi game bao gồm:

- Hệ thống không được cung cấp luật chơi của game. Chính vì thế nó cũng không thể biết được hành động nào nên làm hoặc không nên làm ứng với từng tình huống cụ thể.
- Dữ liệu đầu vào là hình ảnh RGB có kích thước 210×160. Để học được một chiến thuật chơi đơn giản thì hệ thống cũng phải chơi "thử và sai" một số lượng lớn màn chơi (có thể lên đến 10000 frame). Vì vậy, lượng dữ liệu đầu vào cần phải xử lý là rất lớn.
- Các game có hình ảnh, nội dung rất khác nhau. Để có thể học cách chơi của nhiều game khác nhau thì thuật toán học phải mang tính tổng quát cao, không sử dụng các tính chất riêng biệt của từng game.
- Để đạt được điểm số cao (ngang hoặc hơn điểm số của con người) thì phải tìm được chiến thuật chơi mang tính lâu dài. Những phương pháp tham lam, lựa chọn hành động để đạt điểm tối đa trong tương lai gần thường

không tối ưu.

[TODO: Thêm hướng tiếp cận liên quan + các thực nghiệm + Reference]

Trong những năm gần đây, học sâu đạt đươc nhiều bước đột phá trong nhiều lĩnh vực như Thị giác máy tính (Computer Vision), Nhận diện giọng nói (Speech Recognition), ... Việc kết hợp giữa học sâu và học tăng cường đã dẫn đến một hướng tiếp cận mới cho bài toán tự động chơi game: học tăng cường sâu (Deep reinforcement learning) [2]. Với học sâu, ta có thể học được những đặc trưng cấp cao (high level features) từ hình ảnh thô mà không cần phải tự thiết kế đặc trưng bằng tay (hand-designed features). Khi kết hợp với học tăng cường, ta có một hình "**End-to-end**": việc học đặc trưng và học chiến thuật chơi được liên kết chặt chẽ với nhau. Trong luận văn này, chúng em thực hiện việc cài đặt lại phương pháp học tăng cường sâu và thử nghiệm mô hình với những tham số khác nhau. Cùng với đó, luận văn thử nghiệm kỹ thuật học chuyển tiếp (Transfer learning) nhằm giảm thời gian huấn luyện cho nhiều game.

Chương 2

Kiến thức nền tảng

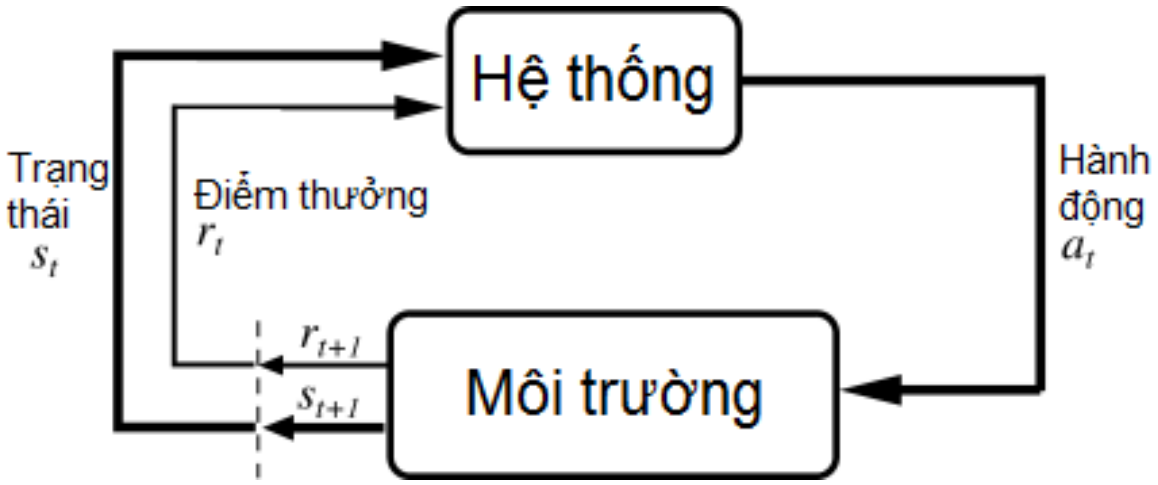
Trong chương này sẽ trình bày những kiến thức nền tảng của học tăng cường. Trong phần đầu tiên chúng em sẽ trình bày định nghĩa của các thành phần cơ bản trong học tăng cường. Tiếp đó sẽ đề cập đến mô hình Markov Decision Processes được áp dụng trong việc đánh giá lý thuyết một số thành phần của bài toán học tăng cường. Cùng với đó sẽ trình bày qui trình tổng quát để đánh giá và cải thiện chính sách trong bài toán. Cuối cùng chúng em sẽ trình bày một số phương pháp phổ biến thường đượcAgent áp dụng để đánh giá cũng như cải thiện giúp hệ thông có cách giải tốt hơn cho bài toán trên.

2.1 Các thành phần cơ bản của học tăng cường

2.1.1 Agent và môi trường

Trong học tăng cường, đối tượng học và đưa ra quyết định được gọi chung là *hệ thống*. Nó tương tác trực tiếp tới một đối tượng được gọi là *môi trường*. Sự tương tác này được diễn ra liên tục. Agent lựa chọn hành động dựa trên những gì nó nhận được từ môi trường. Những thông tin này bao gồm:

- **Trạng thái** (state): Những thông tin hữu ích mà hệ thống có thể cảm nhận được từ môi trường. Ví dụ trong đánh cờ, trạng thái có thể là vị trí những quân cờ đang có trên bàn cờ. Thường được ký hiệu là s.



Hình 2.1: Quá trình tương tác giữa hệ thông và môi trường

- **Điểm thưởng** (reward): Giá trị mà môi trường trả ra tương ứng với trạng thái mà nó vừa đạt được hoặc hành động mà hệ thống vừa thực hiện. Thường được ký hiệu là r. Cũng với ví dụ đánh cờ, điểm thưởng mà hệ thống có thể nhận được từ môi trường trong ví dụ này là: +1 nếu hệ thống thắng, -1 nếu hệ thống thua, và trong quá trình đánh cờ điểm thưởng có thể là 0 cho mỗi trạng thái bàn cờ.

Từ trạng thái và điểm thưởng nhận được, hệ thống dựa vào đó để ra quyết định chọn hành động phù hợp sao cho cố gắng đạt được nhiều điểm thưởng nhất.

Agent và môi trường tương tác theo một chuỗi tuần tự các time-steps, $t = 0, 1, 2, ...$ Tại mỗi time step t, agent nhận những mô tả trạng thái của môi trường, $S_t \in \mathcal{S}$, với $\mathcal{S}$ là tập các trạng thái có thể có. Dựa vào những mô tả trạng thái nhận được, agent chọn một hành động, $A_t \in (S_t)$, trong đó (S_t) là tập các hành động có thể thực hiện tại trạng thái S_t. Tại time step sau đó, agent nhận được giá trị điểm thưởng, $R_{t+1} \in \mathbb{R}$, cùng với trạng thái tiếp theo S_{t+1} Quá trình tương tác giữa agent và môi trường được mô tả trong hình 2.1

Các thành phần của agent gồm có:

- **Chính sách**. Chính sách, π, xác định khả năng chọn một hành động khi hệ thống nhận được một trạng thái s. Chính xác tại time step t được xác định $\pi_t(a \mid s) = \mathbb{P}[A_t = a \mid S_t = s]$. Để đạt được mục tiêu được nhiều điểm thưởng nhất, hệ thống cần có một chính sách chọn lựa hành động phù hợp mỗi khi gặp một trạng thái. Những phương pháp học tăng cường thường

tập trung thay đổi các chính sách của hệ thống sao cho đạt được kết quả tốt trong thực nghiệm.

- **Hàm giá trị**. Hầu hết các thuật toán học tăng cường đầu tập trung đánh giá những hàm giá trị, các hàm này đánh giá một trạng thái hoặc hành động là tốt như thế nào cho agent thông qua việc ước lượng điểm thưởng mà hệ thống có thể nhận được ở tương lai. Thông thường, giá trị của một trạng thái s, dưới một chính sách π được ký hiệu $v_\pi(s)$ là lượng điểm thưởng kỳ vọng nhận được bắt đầu từ trạng thái s về sau.

- **Mô hình**. Trong một số bài toán học tăng cường, hệ thống có thể xây dựng mô hình cho riêng mình để mô phỏng lại môi trường. Qua đó cho phép hệ thống có thể suy luận hoặc dự đoán những thông tin mà nó có thể nhận được từ môi trường trong tương lai.

2.1.2 Returns

Return G_t xác định lượng điểm thưởng mà agent nhận được kể từ thời điểm time step t đến tương lai. Return thường được xác định bằng nhiều hàm khác nhau, trong đó hàm đơn giản nhất xác định return bằng tổng các điểm thưởng có thể nhận được. Nó có dạng như sau:

$$G_t = R_{t+1} + R_{t+2} + ... + R_T \tag{2.1}$$

ở đây T là time step cuối cùng.

Mặt khác, return cũng có thể được xác định bằng tổng điểm thưởng đã bị discount qua từng time step. Nó được định nghĩa như sau:

$$G_t = R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+3} ... + \gamma^{T-1} R_T = \sum_{k=0}^{\infty} \gamma^k R_{t+k+1} \tag{2.2}$$

Trong đó γ là một hệ số với giá trị $0 \leqslant \gamma \leqslant 1$. γ cũng được gọi là tỉ lệ discount. Tỉ lệ này xác định độ tin tưởng của agent vào giá trị điểm thưởng ở tương lai. Khi $\gamma \to 1$, agent có su hướng quan tâm đến giá trị điểm thưởng tương lai càng

nhiều. Đặc biệt với $\gamma = 0$, khi đó agent chỉ quan tâm giá trị điểm thưởng ở hiện tại mà bỏ qua những giá trị điểm thưởng ở tương lai.

2.2 Mô hình Markov Decision Processes

2.2.1 Định nghĩa mô hình Markov Decision Processes

Mô hình Markov Decision Processes (MDP) được sử dụng để mô hình hóa bài toán học tăng cường một cách có hình thức. Cụ thể, MDP là một bộ bao gồm 5 thành phần $< \mathcal{S}, \mathcal{A}, \mathcal{P}, \mathcal{R}, \gamma >$ trong đó:

- $\mathcal{S}$: tập trạng thái hữu hạn có thể có của môi trường.
- $\mathcal{A}$: tập những hành động hữu hạn mà hệ thống có thể thực hiện để tương tác với môi trường.
- γ: Hệ số có giá trị thỏa $0 \leqslant \gamma \leqslant 1$ thể hiện mức độ tin tưởng về giá trị điểm thưởng nhận được ở tương lai.
- $\mathcal{P}$: ma trận xác suất chuyển trạng thái. Trong đó $\mathcal{P}^a_{ss'}$ là xác suất chuyển đến trạng thái s' khi hệ thống đang ở trạng thái s và thực hiện hành động a.

$$\mathcal{P}^a_{ss'} = \mathbb{P}[S_{t+1} = s' \mid S_t = s, A_t = a] \tag{2.3}$$

- $\mathcal{R}$: ma trận điểm thưởng theo từng bộ (trạng thái, hành động). $\mathcal{R}^a_s$ là kỳ vọng giá trị điểm thưởng nhận được khi hệ thống thực hiện hành động a ở trạng thái s.

$$\mathcal{R}^a_s = \mathbb{E}[R_t \mid S_t = s, A_t = a] \tag{2.4}$$

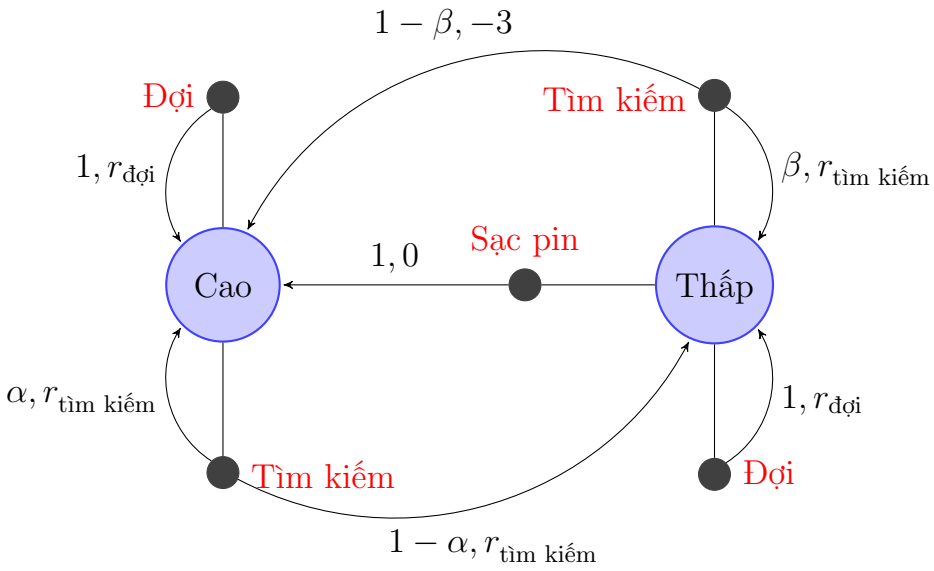
Ví dụ: Mô hình MDP trong robot thu gom Công việc của robot này là thu lượm những lon soda đã được uống hết trong văn phòng. Nó có những cảm biến để xác định những lon soda này, bánh xe và cánh tay để di chuyển và gắp nhặt những lon này bỏ vào thùng. Robot hoạt động bằng pin sạc. Hệ thống điều khiển của robot có chức năng tiếp nhận những thông tin từ cảm biến từ đó điểu khiển bánh xe và cánh tay. Trong ví dụ, chúng em chỉ xét dựa trên mức độ pin

hiện tại robot nên quyết định tìm kiếm những lon soda như thế nào? Robot có thể có ba quyết định (1) thực hiện tìm kiếm một lon soda, (2) đứng yên và đợi người khác mang lon soda đến cho nó, (3) quay trở lại nơi sạc pin. Trạng thái của môi trường được xác định là trạng thái của pin hiện tại của robot. Cách tốt nhất để tìm kiếm những lon soda là robot thực hiện hành động tìm kiếm, nhưng việc này sẽ làm giảm dung lượng của pin. Ngược lại nếu robot đứng yên và đợi thì dung lượng pin của nó không giảm. Mỗi khi dung lượng pin của robot ở mức thấp nó sẽ quay lại chỗ sạc pin. Trường hợp xấu nhất có thể xảy ra là robot không đủ dung lượng pin để quay lại nơi sạc khi đó nó sẽ đứng yên và đợi ai đó mang nó đến chỗ sạc. Do đó robot cần có một chiến lược phù hợp để đạt được hiệu năng cao nhất có thể. Hệ thống đưa ra những quyết định của nó dựa trên mức năng lượng pin. Mức năng lượng này có thể được xác định hai mức *cao* và *thấp*. Khi đó tập trạng thái mà hệ thống có thể nhận được $\mathcal{S} = \{\text{cao}, \text{thấp}\}$. Những hành động của hệ thống trong ví dụ này được xét đơn giản gồm ba hành động *đợi*, *tìm kiếm*, và *sạc pin*. Khi dung lượng pin ở trạng thái cao, hệ thống chỉ thực hiện hai hành động: tìm kiếm và đợi. Ngược lại khi ở trạng thái thấp, hệ thống có thể thực hiện ba hành động: tìm kiếm, đợi, và sạc pin.

$$\mathcal{A}(\text{cao}) = \{\text{tìm kiếm}, \text{đợi}\}$$

$$\mathcal{A}(\text{thấp}) = \{\text{tìm kiếm}, \text{đợi}, \text{sạc pin}\}$$

Khi mức năng lượng pin ở mức cao, việc robot thực hiện tìm kiếm sẽ có xác suất α năng lượng pin vẫn ở mức cao, và $1 - \alpha$ năng lượng của pi sẽ chuyển về mức thấp. Mặt khác, khi mức năng lượng ở mức thấp, nếu robot thực hiện tìm kiếm sẽ có xác suất β năng lượng pin ở mức thấp và $1 - \beta$ chuyển đến mức cao, trường hợp này xảy ra khi dung lượng pin cạn kiệt và cần ai đó mang nó đến chỗ sạc cho đến khi đạt mức năng lượng cao. Ngoài ra, mỗi lần robot thu gom được một lon soda nó sẽ nhận được $+1$ điểm thưởng và sẽ bị -3 điểm thưởng mỗi khi nó phải cần ai đó mang đến chỗ sạc. $r_{\text{đợi}}$, $r_{\text{tìm kiếm}}$ là số lượng lon soda kỳ vọng mà robot có thể thu gom được trong khi đợi và tìm kiếm. Hình 2.2 minh họa cho mô hình MDP trong robot thu gom.



Hình 2.2: Đồ thị minh họa chuyển trạng thái cho robot thu gom. Trong đồ thị có hai loại node: node trạng thái và node hành động. Node trạng thái minh họa những trạng thái có thể có mà hệ thống có thể nhận được, nó được ký hiệu một vòng tròn lớn với tên của trạng thái bên trong. Node hành động tường ứng với cặp (trạng thái, hành động). Việc thực hiện hành động a tại trạng thái s tương ứng trên đồ thị là một cạnh bắt đầu từ node trạng s tới node hành động a. Khi đó môi trường sẽ trả ra trạng thái tiếp theo s' ứng với đích của mũi tên đi từ node hành động a. Xác suất chuyển tới trạng thái s' khi thực hiện hành động a ở trạng thái s $p(s' \mid s, a)$, và giá trị điểm thưởng kỳ vọng nhận được trong trường hợp này $r(s, a, s')$ tương ứng với ký hiệu trên mũi tên. Ví dụ: khi mức năng lượng pin đăng ở trạng thái *thấp*, hệ thống quyết định thực hiện hành động *sạc pin* khi đó trạng thái tiếp theo mà hệ thống nhận được sẽ là mức năng lượng pin ở trạng thái *cao* và xác suất chuyển tới trạng thái *cao* $p(\text{cao} \mid \text{thấp}, \text{sạc pin})$ là 1 và giá trị kỳ vọng điểm thưởng tương ứng $r(\text{thấp}, \text{sạc pin}, \text{cao})$ là 0.

2.2.2 Chính sách và hàm giá trị

Một chính sách π xác định xác suất mà hệ thông thực hiện hành động a khi nó trong trạng thái s được ký hiệu $\pi(a \mid s)$. Có thể nói chính sách như "bộ não" của hệ thống, nó quyết định cách thức mà hệ thống hành động trong những trạng thái cụ thể do đó một chính sách tốt cũng làm cho khả năng hệ thống ra quyết định trở nên tốt hơn.

Hàm giá trị cho biết những trạng thái hoặc những cặp hành động và trạng thái tốt như thế nào cho hệ thông khi nó trong những trạng thái hoặc thực hiện những cặp hành động và trạng thái đó. Khái niệm tốt ở đây nghĩa là giá trị điểm thưởng kỳ vọng mà hệ thông có thể nhận được ở tương lai. Hầu hết các thuật toán trong học tăng cường đều tập trung vào việc đánh giá những hàm giá trị qua đó cải thiện chính sách trở nên tốt hơn. Điểm thưởng mà hệ thống có thể nhận được trong tương lai phụ thuộc vào những hành động mà nó thực hiện. Do đó hàm giá trị chịu ảnh hưởng rất nhiều vào chính sách. Giá trị của trạng thái s dưới một chính sách π, ký hiệu $v_\pi(s)$, là giá trị kỳ vọng của return mà hệ thống nhận được bắt đầu từ trạng thái s theo chính sách π sau đó. Với mô hình MDP, $v_\pi(s)$ được định nghĩa như sau:

$$v_\pi(s) = \mathbb{E}_\pi \left[G_t \mid S_t = s \right] = \mathbb{E}_\pi \left[\sum_{k=0}^{\infty} \gamma^k R_{t+k+1} \middle| S_t = s \right] \tag{2.5}$$

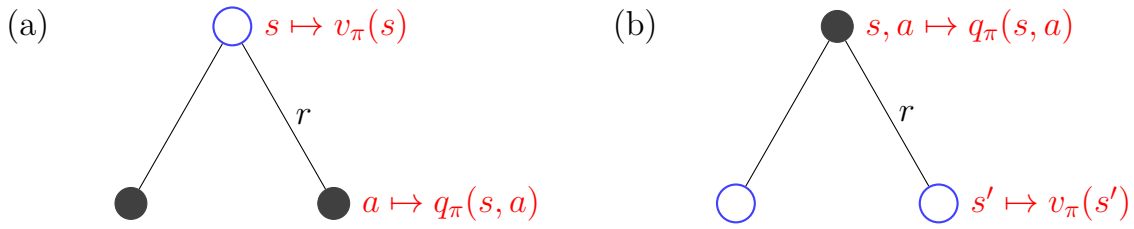
v_π được gọi là hàm giá trị trạng thái dưới chính sách π.

Tương tự, chúng ta định nghĩa giá trị của việc thực hiện hành động a trong trạng thái s dưới chính sách π, được ký hiệu $q_\pi(s, a)$, là giá trị kỳ vọng của return mà hệ thống nhận được bắt đầu từ việc thực hiện hành động a trong trạng thái s theo chính sách π

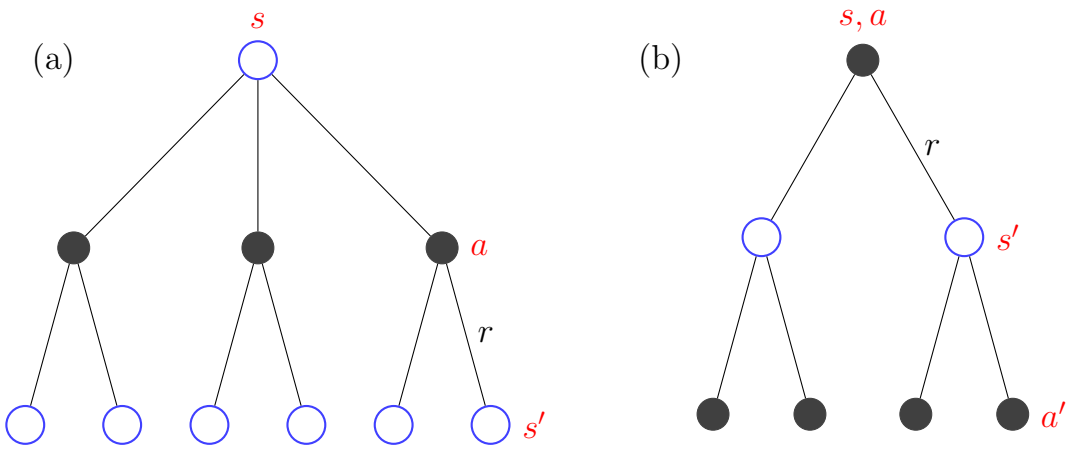
$$q_\pi(s, a) = \mathbb{E}_\pi \left[G_t \mid S_t = s, A_t = a \right] = \mathbb{E}_\pi \left[\sum_{k=0}^{\infty} \gamma^k R_{t+k+1} \middle| S_t = s, A_t = a \right] \tag{2.6}$$

q_π được gọi là hàm giá trị hành động dưới chính sách π.

Hình 2.3 minh họa quan hệ giữa hàm giá trị trạng thái và hàm giá trị hành động, khi có được hàm giá trị này ta có thể có được hàm giá trị còn lại. Phương



Hình 2.3: Đồ thị minh họa quan hệ giữa hàm giá trị trạng thái và hàm giá trị hành động



Hình 2.4: Đồ thị minh họa cho (a) v_π và (b) q_π

trình 2.7 xác hàm giá trị của một trạng thái bằng giá trị kỳ vọng giá trị của các hành động thực hiện tại trạng thái đó. Hình 2.3a minh họa quan hệ giữa giá trị của một trạng thái s và giá trị của các hành động thực hiện tại trạng thái đó. Hình 2.3b cho thấy từ việc thực hiện hành động a tại trạng thái s, môi trường có thể trả ra nhiều trạng thái tiếp theo s' khác nhau. Do đó giá trị của hành động a ở trạng thái s có thể được xác định bằng tổng giá trị kỳ vọng điểm thưởng nhận được và giá trị kỳ vọng của các trạng thái tiếp theo đó đã được nhân với hệ số γ. Cách xác định này được biểu diễn trong phương trình 2.8.

$$v_\pi = \sum_{a \in \mathcal{A}(s)} \pi(a \mid s) q_\pi(s, a) \tag{2.7}$$

$$q_\pi(s, a) = \mathcal{R}_s^a + \gamma \sum_{s' \in \mathcal{S}} \mathcal{P}_{ss'}^a v_\pi(s') \tag{2.8}$$

Hàm giá trị có một tính chất cơ bản thường được áp dụng trong học tăng

cường đó là mối quan hệ đệ quy. Cho bất kỳ chính sách π với bất kỳ trạng thái s, hàm giá trị cho một trạng thái được xác định:

$$\begin{aligned} v_\pi(s) &= \mathbb{E}_\pi \left[G_t \mid S_t = s \right] \\ &= \mathbb{E}_\pi \left[R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+3} + ... \mid S_t = s \right] \\ &= \mathbb{E}_\pi \left[R_{t+1} + \gamma (R_{t+2} + \gamma R_{t+3} + ...) \mid S_t = s \right] \\ &= \mathbb{E}_\pi \left[R_{t+1} + \gamma G_{t+1} \mid S_t = s \right] \\ &= \mathbb{E}_\pi \left[R_{t+1} + \gamma v(S_{t+1}) \mid S_t = s \right] \end{aligned} \tag{2.9}$$

Phương trình 2.9 được gọi là phương trình Bellman cho v_π. Từ phương trình này ta thấy được mối liên quan giữa giá trị của một trạng s bất kỳ và giá trị của những trạng thái tiếp theo đạt được từ trạng thái đó. Ý tưởng nhìn trước một bước, hay nói cách khác đánh giá trạng thái hiện tại bằng cách nhìn trước tất cả những trạng tái tiếp theo có thể đạt được từ trạng thái đó, được minh họa trong hình 2.4a. Từ một trạng thái, môi trường có thể trả ra nhiều điểm thưởng r và trạng thái tiếp theo s' khác nhau. Phương trình 2.9 sẽ trung bình tất cả các trường hợp có thể đó lại theo xác suất mà chúng xuất hiện. Phương trình này cũng cho thấy giá trị của một trạng thái phải bằng tổng giá trị kỳ vọng của những trạng thái tiếp sau đó và giá trị kỳ vọng điểm thưởng nhận được.

$$q_\pi(s, a) = \mathbb{E}_\pi \left[R_{t+1} + \gamma q_\pi(S_{t+1}, A_{t+1}) \mid S_t = s, A_t = a \right] \tag{2.10}$$

Phân tích phương trình 2.6 tương tự như đã làm đối với hàm giá trị hành động, ta có được phương trình 2.10. Hình 2.4b minh họa ý tưởng nhìn trước một đước để đánh giá giá trị của một hành động ở trạng thái hiện tại. Từ một hành động a ở trạng thái s, môi trường có thể trả ra nhiều điểm thưởng r và trạng thái s' khác nhau. Trong mỗi trạng thái s' lại có nhiều hành động a' khác nhau có thể thực hiện. Phương trình 2.10 sẽ trung bình tất cả các trường hợp có thể đó lại theo xác suất mà chúng được thực hiện. Hay nói cách khác, phương trình 2.10 cho thấy giá trị của một hành động a tại trạng thái s, $q_\pi(a, s)$ cũng được xác định tổng bằng giá trị kỳ vọng điểm thưởng hệ thống nhận được nhận được ngay sau khi thực hiện thực hiện hành động đó và giá trị kỳ vọng của các hành

động trong những trạng thái kế tiếp.

2.2.3 Hàm giá trị tối ưu

Để giải quyết những vấn đề trong học tăng cường, chúng ta cần tìm một chính sách sao cho hệ thống có thể đạt được nhiều điểm thưởng nhất có thể. Một chính sách π được xác định là tốt hơn hoặc bằng chính sách π' khi giá trị kỳ vọng của return theo chính sách π lớn hơn hoặc bằng giá trị đó theo chính sách π'. Hay có thể định nghĩa theo cách khác:

$$\pi \geq \pi' \iff v_\pi(s) \geq v_{\pi'}(s), \forall s \in \mathcal{S} \tag{2.11}$$

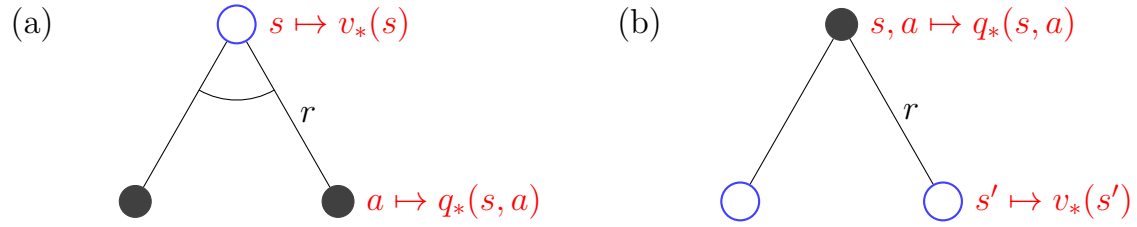
Luôn có ít nhất một chính sách tốt hơn hoặc bằng tất cả các chính sách còn lại [3]. Chúng được gọi chung là *chính sách tối ưu* và được ký hiệu π_*. Những chính sách tối ưu đều cùng có chung một hàm giá trị trạng thái và hàm giá trị hành động. Hai loại hàm giá trị này có thể được gọi chung là *hàm giá trị tối ưu*. Chúng ta cũng có thể gọi tách biệt *hàm giá trị trạng thái tối ưu* đối với hàm giá trị trạng thái và *hàm giá trị hành động tối ưu* đối với hàm giá trị hành động. Phương trình 2.12 và 2.13 định nghĩa hình thức cho hai loại hàm này

$$v_*(s) = \max_\pi v_\pi(s), \forall s \in \mathcal{S} \tag{2.12}$$

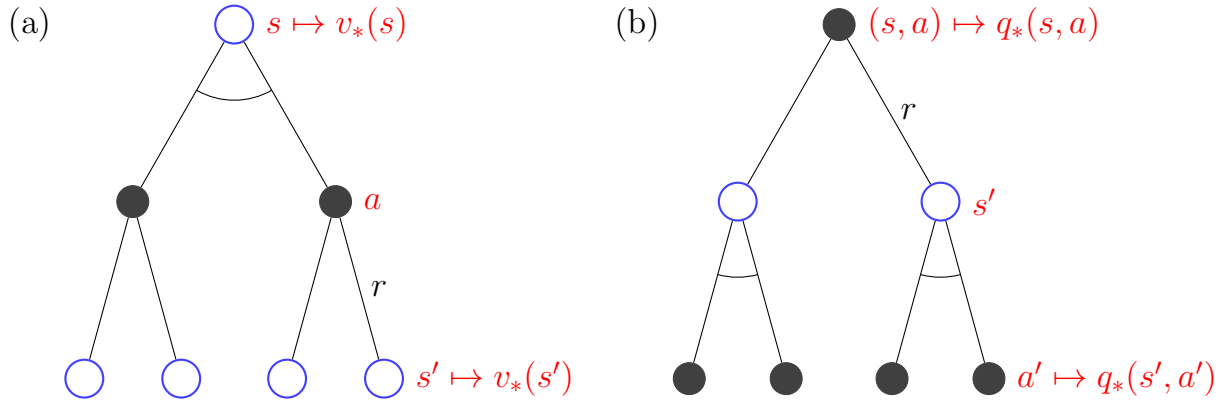
$$q_*(s,a) = \max_\pi q_\pi(s,a), \forall s \in \mathcal{S} \text{ và } \forall a \in \mathcal{A}(s) \tag{2.13}$$

Từ hai phương trình 2.12 và 2.13 thấy rằng để xác định hàm giá trị tối ưu của mỗi trạng thái s hoặc cặp trạng thái và hành động (s,a), ta cần thử đánh giá giá trị của chúng theo tất cả các chính sách có thể có và chọn giá trị cao nhất là giá trị tối ưu cho trạng thái s hoặc cặp trạng thái và hành động (s,a).

Hình 2.5 minh họa quan hệ giữa giá trị trạng thái tối ưu và hàm giá trị hành động tối ưu, khi có được hàm này ta dễ dàng có được hàm còn lại. Trong hình 2.5a, ta có thể xác định giá trị tối ưu cho trạng thái s dựa trên hàm giá trị hành động tối ưu của các hành động có thể thực hiện tại trạng thái đó. Phương trình 2.14 xác định giá trị tối ưu cho trạng thái s bằng cách chọn giá trị hành động tối ưu lớn nhất trong các hành động có thể thực hiện ở trọng thái đó. Tương tự



Hình 2.5: Đồ thị minh họa quan hệ giữa hàm giá trị trạng thái tối ưu và hàm giá trị hành động tối ưu



Hình 2.6: Đồ thị minh họa phương trình Bellman trong (a) v_* và (b) q_*

trong hình 2.5b, ta có thể xác định giá trị tối ưu cho hành động a ở trạng thái s, dựa trên hàm giá trị trạng thái tối ưu của các trạng thái kế tiếp đạt được từ hành động đó. Phương trình 2.15 xác định giá trị tối ưu của hành động a tại trạng thái s bằng tổng giá trị kỳ vọng điểm thưởng nhận được từ môi trường và giá trị tối ưu kỳ vọng của những trạng thái kế tiếp đã nhân với hệ số γ.

$$v_*(s) = \max_{a \in \mathcal{A}(s)} q_*(s,a) \tag{2.14}$$

$$q_*(s,a) = R_s^a + \gamma \sum_{s' \in \mathcal{S}} \mathcal{P}_{ss'}^a v_*(s') \tag{2.15}$$

$$v_*(s) = \max_{a \in \mathcal{A}(s)} R_s^a + \gamma \sum_{s' \in \mathcal{S}} \mathcal{P}_{ss'}^a v_*(s') \tag{2.16}$$

$$q_*(s,a) = R_s^a + \gamma \sum_{s' \in \mathcal{S}} \mathcal{P}_{ss'}^a \max_{a' \in \mathcal{A}(s')} q_*(s',a') \tag{2.17}$$

Phương trình 2.16 và 2.17 dễ dàng có được bằng cách thay thế hai phương trình 2.14 và 2.15 qua lại lẫn nhau. Từ hai phương trình này, ta thấy được dạng phương trình Bellman trong hàm giá trị trạng thái tối ưu và hàm giá trị hành động tối ưu. Hình 2.6 minh họa ý tưởng nhìn trước một bước của phương trình Bellman trong hàm giá trị tối ưu. Trong đó hình 2.6a minh họa cách thức xác định giá trị tối ưu cho một trạng thái ứng với phương trình 2.16. Hình 2.6b minh họa cách thức xác định giá trị tối ưu của một hành động ở một trạng thái ứng với phương trình 2.17.

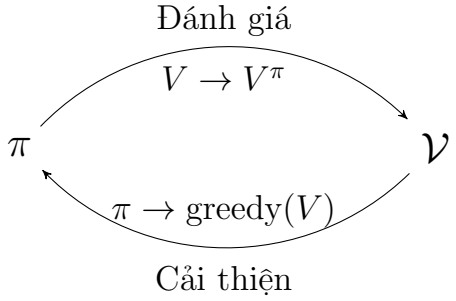
2.3 Quy trình lặp chính sách

Trong các bài toán học tăng cường, mục tiêu chính của ta là tìm được chính sách tối ưu π_* nhằm giúp cho hệ thông giải quyết bài toán tốt nhất có thể. Do đó ta cần có quy trình để thay đổi chính sách hiện tại trở nên tối ưu. Quy trình này được gọi là *quy trình lặp chính sách*. Hình 2.7 minh họa quy trình chung của lặp chính sách. Trong quy trình này được chia thành hai giai đoạn:

- **Đánh giá chính sách**: Việc đánh giá một chính sách π được thực hiện bằng cách xác định hàm giá trị trạng thái của dưới chính sách đó.
- **Cải thiện chính sách**: Sau khi có được hàm giá trị của một chính sách π, chính sách cải thiện mới π' được tạo ra bằng cách thực hiện tham lam trên hàm giá trị của chính sách π, tức là chỉ chọn thực hiện hành động có giá trị cao nhất dựa trên hàm giá trị trạng thái; việc này có thể thực hiện được do mối quan hệ giữa hai loại hàm.

Một chính sách π_1 được cải thiện từ chính sách π_0 dựa trên hàm giá trị trạng thái v_{π_0}. Khi có được chính sách π_1 ta có thể tính được hàm giá trị v_{π_1} qua đó tiếp tục cải thiện để có được chính sách π_2. Quá trình này diễn ra cho đến khi đạt được chính sách tối ưu.

$$\pi_0 \xrightarrow{\text{Đánh giá}} v_{\pi_0} \xrightarrow{\text{Cải thiện}} \pi_1 \xrightarrow{\text{Đánh giá}} v_{\pi_1} \xrightarrow{\text{Cải thiện}} \pi_2 \cdots \xrightarrow{\text{Cải thiện}} \pi_* \xrightarrow{\text{Đánh giá}} v_{\pi_*}$$



Hình 2.7: Quy trình chung trong lặp chính sách

2.3.1 Phương pháp đánh giá chính sách

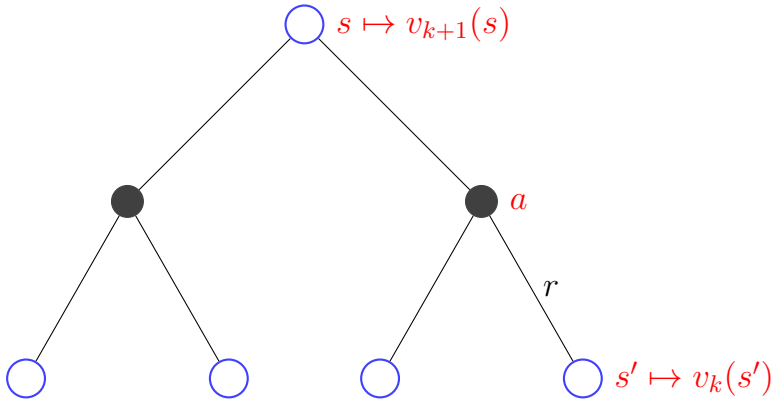
Trong phần này, chúng em sẽ trình bày một số phương pháp được áp dụng để đánh giá chính sách.

Quy hoạch động (Dynamic Programming)

Quy hoạch động thường được dùng để giải quyết các bài toán tối ưu mà dữ liệu có tính thứ tự, ví dụ như dữ liệu chuỗi hay dữ liệu thời gian. Một bài toán tối ưu có thể được giải quyết bằng quy hoạch động cần có hai đặc điểm:

- Quy tặc tối ưu (Principle of Optimality): các bài toán có thể phân rã thành các bài toán con, và kết quả của bài toán con này đóng góp vào lời giải của bài toán gốc.
- Các bài toán con chồng lấn lên nhau và lặp lại nhiều lần: nhằm tận dụng lại kết quả của những bài toán con đã tính toán trước đó.

Trong nhiều bài toán học tăng cường, kỹ thuật quy hoạch động được dùng để tìm chính sách tối ưu hoặc tối ưu hàm giá trị. Để có thể áp dụng kỹ thuật quy hoạch động, những bài toán này cũng cần phải thỏa yêu cầu là hệ thống có kiến thức đầy đủ về môi trường hay cách khác môi trường có mô hình MDP. Quy hoạch động xác định hàm giá trị của một chính sách bằng cách cập nhật hàm giá trị được khởi tạo bất kỳ ban đầu qua nhiều vòng lặp, dựa vào phương trình Bellman. Ý tưởng của cách xác định này như sau: Ban đầu khởi tạo hàm giá trị v_0 bất kỳ cho tất cả các trạng thái, trừ trạng thái kết thúc được luôn có giá trị là 0. Tiến hành cập nhật hàm giá trị mới v_1 cho chính sách dựa trên hàm



Hình 2.8: Đồ thị minh họa cập nhật hàm giá trị bằng quy hoạch động

giá trị v_0 theo phương trình 2.18. Tương tự cập nhật hàm giá trị mới v_2 dựa trên v_1. Quá trình lặp cho đến khi độ khác biệt giữa hàm giá trị sau và giá trị trước đó nhỏ hơn một lượng cho trước. Quy trình cập nhật được minh họa trong hình 2.8, trong đó giá trị mới v_{k+1} của trạng thái s được xác định dựa trên giá trị kỳ vọng điểm thưởng nhận được theo chính sách π, và giá trị hiện tại v_k của các trạng thái s' kế tiếp trạng thái s'. Tổng thể của việc đánh giá chính xác theo quy hoạch động được trình bày ở thuật toán 2.1

$$v_{k+1}(s) \leftarrow \sum_{a \in \mathcal{A}} \pi(a \mid s)(\mathcal{R}_s^a + \gamma \sum_{s' \in \mathcal{S}} \mathcal{P}_{ss'}^a v_k(s')) \tag{2.18}$$

[TODO] Ví dụ minh họa TD

Mặc dù đã quy hoạch động đã được chứng minh là xấp xỉ tốt hay thậm chí là tìm được hàm giá trị trạng thái của chính sách π [1], nhưng trong các bài toán học thực tế của học tăng cường đặc biệt là những bài toán lớn thì quy hoạch động trở nên không khả thi do chi phí tính toán cao, trong trường hợp xấu nhất chi phí tính toán thuộc $O(k^n)$ với k là số hành động và n là số trạng thái. Ngoài ra, trong nhiều bài toán thực tế thông thường chúng ta không có kiến thức đầy đủ về môi trường như ma trận chuyển trạng thái $\mathcal{P}$, ma trận điểm thưởng $\mathcal{R}$, tập các trạng thái $\mathcal{A}$. Do đó hệ thống phải có khả năng học từ những thông tin mà nó tiếp nhận được qua việc tương tác với môi trường. Các thông tin này thường ở dạng chuỗi (trạng thái, hành động, điểm thưởng) $S_1, A_1, R_2, S_2, A_2, R_3, \ldots, S_T$. Với những đặc điểm đó, quy hoạch động không thể áp dụng để đánh giá chính

Thuật toán 2.1 Đánh giá hàm giá trị theo quy hoạch động

Đầu vào: Chính sách π cần đánh giá
Đầu ra: Hàm giá trị V xấp xỉ hàm giá trị v_π của chính sách π
Thao tác:

Khởi tạo ngẫu nhiên $V(s)$ cho tất cả trạng thái s không phải trạng thái kết thúc. Nếu s là trạng thái kết thúc, $V(s) = 0$
repeat
    $\Delta \leftarrow 0$ %% Tính độ khác biệt giữa hàm giá trị cũ và giá trị mới. Độ lớn của Δ được xác định là độ khác biệt lớn nhất giữa giá trị cũ và giá trị mới của một trạng thái trong tất cả các trạng thái.
    for $s \in \mathcal{S}$ **do** %% Với mỗi trạng thái
        $v \leftarrow V(s)$ %% Lưu giá trị hiện tại của trạng thái s
        $V(s) \leftarrow \sum_{a \in \mathcal{A}} \pi(a \mid s)(\mathcal{R}_s^a + \gamma \sum_{s' \in \mathcal{S}} \mathcal{P}_{ss'}^a V(s'))$ %% Tính giá trị mới cho trạng thái s dựa trên giá trị hiện tại của các trạng thái s' kế tiếp của trạng thái s, và giá trị kỳ vọng của các hành động tại trạng thái đó theo chính sách π.
        $\Delta \leftarrow \max(\Delta, |v - V(s)|)$ %% Cập nhật giá trị mới cho Δ
    end for
until $\Delta < \theta$ (Một lượng đủ nhỏ)

sách trong các bài toán này.

Monte Carlo (MC)

Tương tự với quy hoạch động, Monte Carlo (MC) xác định hàm giá trị của một chính sách bằng cách cập nhật hàm giá trị khởi tạo qua nhiều vòng lặp. Điểm biệt khác với quy hoạch động của phương pháp MC là nó có thể áp dụng để đánh giá chính sách khi hệ thông không có kiến thức đầy đủ về môi trường. MC dựa trên những thông tin mà hệ thông có được qua việc tương tác với môi trường để xấp xỉ hàm giá trị. Thông thường những thông tin này được chia thành các *episode*. Mỗi episode là một chuỗi bắt đầu từ một trạng thái bất kỳ cho đến khi đạt được một trong những trạng thái kết thúc. Khi đó MC chỉ thực hiện cập nhật hàm giá trị khi kết thúc một episode.

Ý tưởng của MC để đánh giá giá trị của một trạng thái s dựa trên các mẫu thực nghiệm. MC xác định giá trị của trạng thái s bằng cách trung bình những return mà hệ thông nhận được sau khi hệ thống quan sát được trạng thái s. Khi quan sát càng nhiều mẫu thực nghiệm xuất hiện trạng thái s, giá trị trung bình

sẽ càng xấp xỉ tốt giá trị thực của trạng thái này theo chính sách π.

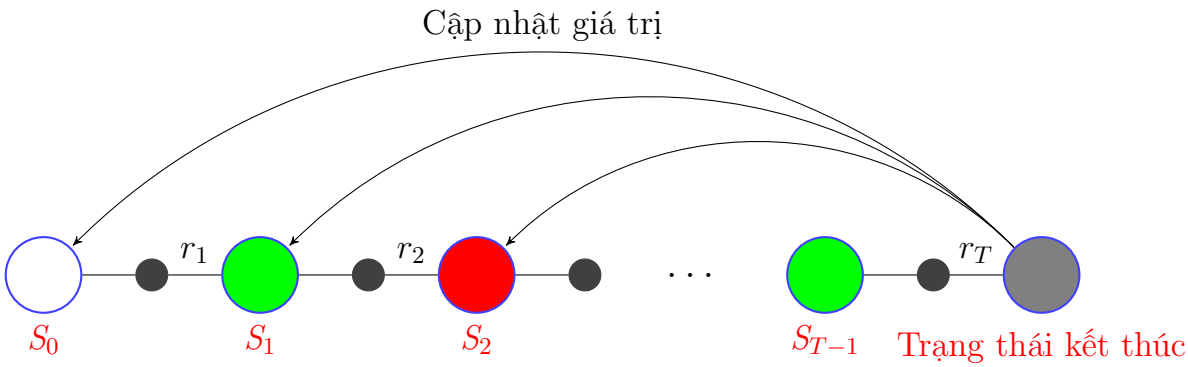
Một episode đạt được bằng cách thực hiện theo chính sách π. Giá trị của trạng thái s $v(s)$ được tính dựa trên những episode có trạng thái s xuất hiện. Một trạng thái s có thể xuất hiện nhiều lần trong một episode. Lần xuất hiện đầu tiên của trạng thái s trong một episode được gọi là first-visit trạng thái đó. Phương pháp first-visit MC xác định giá trị trạng thái s $v_\pi(s)$ bằng trung bình tất cả return mà hệ thống nhận sau lần first-visit của trạng thái s trong các episode. Tổng thể của việc đánh giá chính sách bằng first-visit MC được trình bày ở thuật toán 2.2. Hình 2.9 minh họa cách thức cập nhật hàm giá trị trên một episode.

Thuật toán 2.2 Đánh giá hàm giá trị trạng thái bằng phương pháp first-visit MC

Đầu vào: Chính sách π cần đánh giá
Đầu ra: Hàm giá trị V xấp xỉ hàm giá trị v_π của chính sách π
Thao tác:

Khởi tạo ngẫu nhiên $V(s)$ cho tất cả trạng thái s không phải trạng thái kết thúc. Nếu s là trạng thái kết thúc, $V(s) = 0$
Khởi tạo danh sách rỗng **Returns(**s**)** cho tất cả trạng thái $s \in \mathcal{S}$ %% Danh sách Returns(s) chứa tất cả các return mà hệ thống nhận được sau lần first-visit của trạng thái s trong các episode.
repeat
    Tạo một episode E bằng chính sách π
    for mỗi trạng thái s xuất hiện lần đầu trong E **do**
        $G \leftarrow$ return nhận được sau lần xuất hiện đầu tiên của s
        Thêm G vào danh sách Returns(s)
        $V(s) \leftarrow$ average(Returns(s))
    end for
until Thỏa điều kiện dừng

Trong nhiều trường hợp, hệ thống không có được mô hình của môi trường, việc đánh giá hàm giá trị hành động trở nên khả thi hơn hàm giá trị trạng thái. Với việc có được mô hình của môi trường, hàm giá trị trạng thái là đủ để cải thiện một chính xách trở nên tốt hơn; nó đơn giản là nhìn trước trạng thái tiếp theo và chọn bất kỳ hành động nào dẫn đến trạng thái đó mà đạt được nhiều điểm thưởng nhất. Ngược lại, nếu không có được mô hình của môi trường, hàm giá trị trạng thái là không đủ do hệ thống không thể xác định được trạng thái



Hình 2.9: Đồ thị minh họa cập nhật hàm giá trị trên một episode bằng phương pháp first-visit MC. Hình tròn lớn được ký cho trạng thái xuất hiện. Hình tròn nhỏ được ký cho hành động thực hiện. Màu xác khác nhau giữa các hình tròn biểu thị cho sự khác nhau giữa các trạng thái. Phương pháp first-visit MC chỉ cập nhật giá trị cho các trạng thái khi kết thúc một episode, và mỗi trạng thái chỉ được cập nhật một lần mặc dù trạng thái đó có thể xuất hiện nhiều lần trong một episode

tiếp theo là gì. Vì vậy, nó cần đánh giá giá trị của mỗi hành động trong mỗi trạng thái để xác định hành động nào nên thực hiện ở mỗi trạng thái qua đó cải thiện chính sách đang thực hiện. Để hàm giá trị q_π được thực hiện tương tự như đã hàm giá trị hành động v_π. Để đánh giá giá trị của hành động a tại trạng thái s, nó thực hiện tính trung bình các return mà hệ thộng nhận được dựa vào các episode có sự xuất hiện của cặp trạng thái hành động (s, a). Lẫn xuất hiện đầu tiên của cặp trạng thái và hành động (s, a) trong một episode được gọi là first-visit cặp trạng thái và hành động đó. Phương pháp first-visit MC xác định giá trị của hành động a ở trạng thái s, $q_\pi(s, a)$ bằng trung bình tất cả các return nhận được sau lần first-visit của cặp (s, a) trong các episode. Thuật toán 2.3 trình bày cách thức đánh giá hàm giá trị hành động bằng first-visit MC.

Temporal Difference (TD)

Sarsa

Q-Learning

2.3.2 Phương pháp cải thiện chính sách

Thuật toán 2.3 Đánh giá hàm giá trị hành động bằng phương pháp first-visit MC

Đầu vào: Chính sách π cần đánh giá
Đầu ra: Hàm giá trị V xấp xỉ hàm giá trị v_π của chính sách π
Thao tác:

Khởi tạo ngẫu nhiên $Q(s, a)$ cho tất cả các cặp trạng thái, hành động s, a.
Khởi tạo danh sách rỗng **Returns**(s, a) cho tất cả các cặp trạng thái, hành động (s, a). %% Danh sách Returns(s, a) chứa tất cả các return mà hệ thống nhận được sau lần first-visit của cặp trạng thái, hành động (s, a) trong các episode.
repeat
    Tạo một episode E bằng chính sách π
    for mỗi cặp trạng thái, hành động (s, a) xuất hiện lần đầu trong E **do**
        $G \leftarrow$ return nhận được sau lần xuất hiện đầu tiên của cặp trạng thái, hành động (s, a)
        Thêm G vào danh sách Returns(s, a)
        $Q(s, a) \leftarrow$ average(Returns(s))
    end for
until Thỏa điều kiện dừng

Chương 3

Kết hợp học sâu với học tăng cường

Học sâu đạt được nhiều kết quả khả quan trong nhiều lĩnh vực như xử lý ngôn ngữ tự nhiên, nhận diện giọng nói, nhận dạng đối tượng ... Ưu điểm của học sâu là xấp xỉ hàm tốt, rút trích đặc trưng ở mức high-level.

Trong chương này sẽ trình bày chi tiết những phương pháp mà chúng em áp dụng để giải quyết bài toán "tự động chơi game". Chúng em sử dụng phương pháp Q-Learning, một phương pháp đánh giá được sử dụng phổ biến trong nhiều công trình nghiên cứu về học tăng cường gần đây, để đánh giá hàm giá trị của hành động cho bài toán này. Ngoài ra, chúng em cũng thực hiện xấp xỉ hàm giá trị của bài toán bằng mạng nơ-ron và áp dụng hai kỹ thuật Experience Replay và Fixed Q-targets để tăng tính ổn định cho quá trình học.

Chương 4

Kết quả thực nghiệm

Trong chương này sẽ trình bày chi tiết về cấu trúc mô hình mà chúng em đã thiết lập. Đồng thời đề cập đến những phương pháp để đánh giá mô hình học và những kết quả thực nghiệm đã nhận được. Qua đó so sánh với các phương pháp đã đề xuất trước đây để thấy được tính hiệu quả của mô hình này.

4.1 Giới thiệu Arcade Learning Environment

4.2 Giới thiệu cấu trúc mạng và các siêu tham số đã chọn

4.3 Kết quả thực nghiệm

Chương 5

Kết luận và hướng phát triển

TÀI LIỆU THAM KHẢO

[1] G. J. Gordon, "Stable function approximation in dynamic programming," in *Proceedings of the twelfth international conference on machine learning*, 1995, pp. 261–268. 18

[2] V. Mnih, K. Kavukcuoglu, D. Silver, A. A. Rusu, J. Veness, M. G. Bellemare, A. Graves, M. Riedmiller, A. K. Fidjeland, G. Ostrovski, S. Petersen, C. Beattie, A. Sadik, I. Antonoglou, H. King, D. Kumaran, D. Wierstra, S. Legg, and D. Hassabis, "Human-level control through deep reinforcement learning," *Nature*, pp. 529–533, 2015. 4

[3] R. S. Sutton and A. G. Barto, *Introduction to reinforcement learning*. MIT Press Cambridge, 1998, vol. 135. 14