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

2.3 Những phương pháp đánh giá và cải thiện chính sách 11

Chương 3 Kết hợp học sâu với học tăng cường 12

3.1 Học sâu 12

3.1.1 Khó khăn của bài toán tự động chơi game 12

3.2 Học tăng cường sâu 14

Chương 4 Kết quả thực nghiệm 15

4.1 Giới thiệu Arcade Learning Environment 15

4.2 Giới thiệu cấu trúc mạng và các siêu tham số đã chọn 15

4.3 Kết quả thực nghiệm 15

Chương 5 Kết luận và hướng phát triển **16**

TÀI LIỆU THAM KHẢO **17**

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

DANH MỤC BẢNG

Chương 1

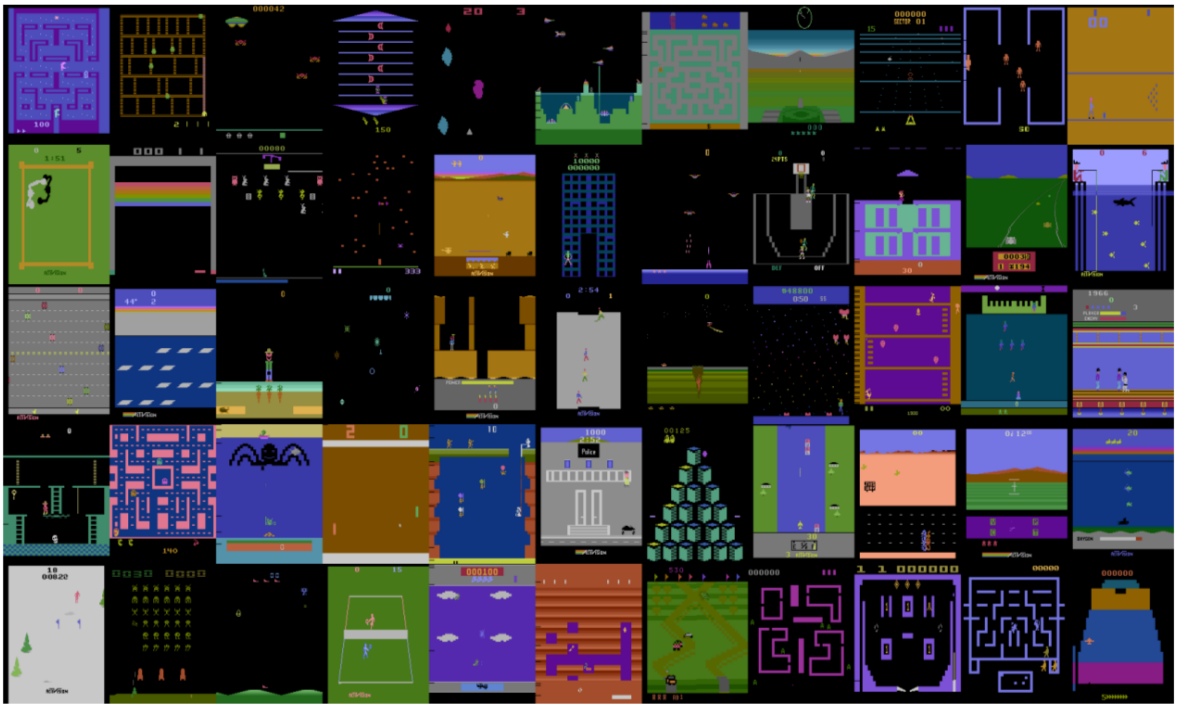
Giới thiệu

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) [1]. 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

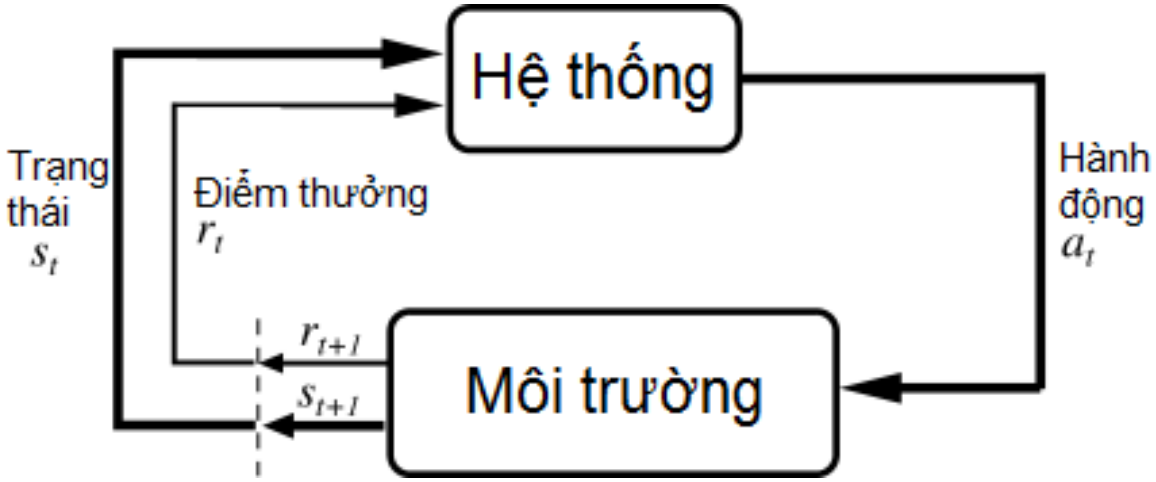
Kiến thức nền tảng

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 *agent*. 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. Môi trường cung cấp giá trị điểm thưởng (reward) cho hành động vừa được thực hiện và những quan sát (observation) tiếp theo cho agent. Từ những quan sát này, agent có thể xây dựng ra các *trạng thái* (state) dựa vào đó để ra quyết định chọn hành động với mục tiêu cố gắng đạt được nhiều điểm thưởng nhất.



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

Cụ thể hơn, 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 agent nhận được một trạng thái s. Chính xác tại time step t được xác định $\pi_t(a|s) = \mathbb{P}[A_t = a|S_t = s]$. Để đạt được mục tiêu được nhiều điểm thưởng nhất, agent 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 agent 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 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**. Agent xây dựng mô hình cho riêng mình để mô phỏng môi trường và dự doán các thông tin của 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:

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

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:

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

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.

Trong thực nghiệm, việc tương tác giữa agent và môi trường có thể được phân chia thành những chuỗi con. Chúng được gọi là những *episode*. [TODO]

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.

$$\mathcal{P}^a_{ss'} = \mathbb{P}[S_{t+1} = s' | S_t = s, A_t = a] \tag{2.1}$$

- $\mathcal{R}$: 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.

$$\mathcal{R}^a_s = \mathbb{E}[R_t | S_t = s, A_t = a] \tag{2.2}$$

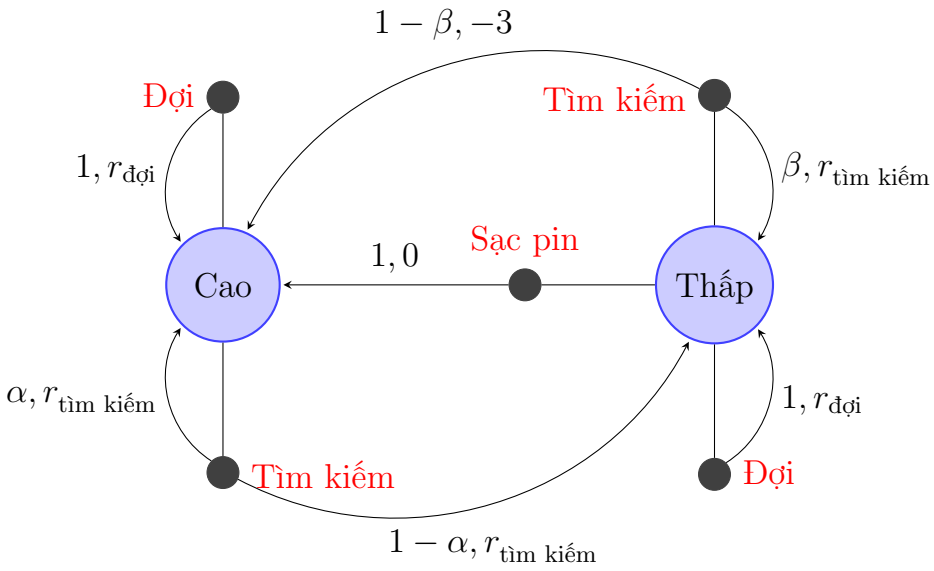
Ví dụ: Mô hình MDP trong robot thu gom Công việc của robot này là thu lượm những lon soda đã được uống hết trong văn phòng. Nó có những cảm biến để xác định những lon soda này, bánh xe và cánh tay để di chuyển và gắp nhặt những lon này bỏ vào thùng. Robot hoạt động bằng pin sạc. Hệ thống điều khiển của 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. Đồ thị 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 $p(s'|s,a)$, 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* $p(\text{cao}|\text{thấp}, \text{sạc pin})$ là 1 và giá trị kỳ vọng điểm thưởng tương ứng $r(\text{thấp}, \text{sạc pin}, \text{cao})$ là 0.

2.3 Những phương pháp đánh giá và cải thiện chính sách

- Dẫn nhập: Trên thực tế ta không có thông tin về môi trường
- Qui trình đánh giá chính sách
  + Dựa trên hàm giá trị trạng thái: Monte-Carlo, TD(0), n-step TD, TD(λ)
  + Dựa trên hàm giá trị hành động: Monte-Carlo, Sarsa(0), n-step Sarsa, Sarsa(λ)
- Qui trình cải thiện chính sách: Phương pháp greedy

Chương 3

Kết hợp học sâu với học tăng cường

Những thành công gần đây của học sâu (Deep learning) trong các bài toán như xử lý ngôn ngữ tự nhiên, nhận diện đối tượng trong ảnh... đặt ra vấn đề: liệu các kỹ thuật trong học sâu có thể áp dụng vào học tăng cường? Để trả lời câu hỏi đó, chương này trình bày về hướng tiếp cận kết hợp học sâu với học tăng cường để áp dụng vào bài toán "Tự động chơi game". Hướng tiếp cận mới mẻ này của lĩnh vực học tăng cường này mang tên "Học tăng cường sâu" (Deep reinforcement learning)
Chương này trình bày hai phần:

- *Nguyên nhân cần sử dụng học sâu và kiến thức cơ bản về học sâu*
- *Áp dụng học tăng cường sâu vào bài toán "Tự động chơi game"*

3.1 Học sâu

3.1.1 Khó khăn của bài toán tự động chơi game

Những thuật toán học tăng cường được trình bày trong chương trước đều tìm chính sách tối ưu dựa vào hàm giá trị. Việc tính **đúng** và **nhanh** hàm giá trị ảnh hưởng rất nhiều đến kết quả của bài toán. Các thuật toán học tăng cường

cổ điển như "Monte Carlo" (MC) hay "Temporal-Difference" (TD) đều đã được chính minh là luôn hội tụ trong những điều kiện nhất định [2]. Ngoài ra, khi áp dụng vào các bài toán kinh điển của học tăng cường thì các thuật toán này đều hội tụ khá nhanh.

Tuy nhiên, với những bài toán thực tế với số trạng thái rất lớn thì việc lưu véc-tơ hàm giá trị trạng thái v_π (hoặc ma trận hàm giá trị hành động q_π) là việc không thể. Ví dụ như "frame hình" của bài toán tự động chơi game có kích thước $210 \times 160 \times 3 = 100800$ điểm ảnh; mỗi điểm ảnh có giá trị trong khoảng $[0, 127]$ nên số trạng thái có thể có lên đến 128^{100800}. Vì vậy, việc lưu trữ hàm giá trị dưới dạng bảng là không khả thi về mặt bộ nhớ. Còn về mặt tốc độ tính toán thì các thuật toán học tăng cường trên đều tính hàm giá trị *rời rạc* cho từng trạng thái. Với số trạng thái quá lớn như trên thì ta không thể duyệt lần lượt từng trạng thái để tính được.

Những lý do trên dẫn đến việc sử dụng một phương pháp xấp xỉ hàm là bắt buộc cho các bài toán học tăng cường với số trạng thái lớn. Một trong những tiếp cận rất tự nhiên đó là sử dụng các mô hình học có giám sát như là một phương pháp xấp xỉ hàm giá trị. Đặc biệt, với những đột phá gần đây của học sâu trong lĩnh vực xử lý ảnh, video... thì việc áp dụng các mô hình học như mạng nơ-ron "Neural networks", "Convolutional Neural Networks" (CNN) vào bài toán tự động chơi game là rất khả quan.

Mạng nơ-ron là một công cụ xấp xỉ hàm giá trị rất linh hoạt với khả năng xấp xỉ hàm đích bất kỳ. Khi đó, chi phí lưu trữ hàm giá trị được giảm đáng kể: ta chỉ cần lưu bộ trọng số của mạng nơ-ron (thông thường chỉ khoảng vài triệu đến vài trăm triệu tham số). Nhưng một trong những tính chất quan trọng nhất của mạng nơ-ron chính là **khả năng tổng quát hoá** (Generalization). Nhờ đặc điểm này mà quá trình học tăng cường được tăng tốc rất đáng kể. Thay vì phải duyệt qua từng trạng thái (thậm chí phải duyệt nhiều lần) để tính hàm giá trị tại đó, mạng nơ-ron có khả năng "dự đoán" rất tốt giá trị của một trạng thái chưa từng thấy dựa vào những trạng thái đã thấy. Như trong bài toán tự động chơi game thì các "frame hình" liên tiếp thường rất giống nhau và các trạng thái này thường cũng có giá trị tương đương nhau.

3.2 Học tăng cường sâu

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] 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

[2] R. S. Sutton and A. G. Barto, *Reinforcement learning: An introduction*. MIT press, 1998. 13