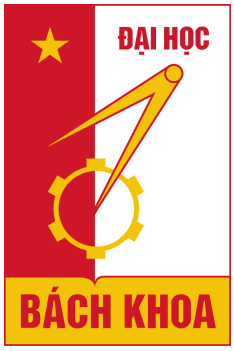
TRƯỜNG ĐẠI HỌC BÁCH KHOA HÀ NỘI
VIỆN TOÁN ỨNG DỤNG VÀ TIN HỌC



BÁO CÁO MÔN HỌC
CÁC MÔ HÌNH NGẪU NHIÊN VÀ ỨNG DỤNG

ĐỀ TÀI: HỌC TĂNG CƯỜNG

Giảng viên hướng dẫn: **TS. NGUYỄN THỊ NGỌC ANH**

Nhóm sinh viên thực hiện:

PHÙNG TRỌNG HIẾU	**20151366**
CAO ĐĂNG SAO	**20163476**
NINH NGỌC LUYÊN	**20142747**
NGUYỄN MẠNH CƯỜNG	**20140596**

HÀ NỘI - 2020

Danh mục hình vẽ

Lời cảm ơn

Để hoàn thành được báo cáo môn học: "Các mô hình ngẫu nhiên và ứng dụng", lời đầu tiên chúng em xin chân thành cảm ơn cô giáo hướng dẫn TS. Nguyễn Thị Ngọc Anh đã truyền đạt những kinh nghiệm, kỹ năng, kiến thức nền tảng để giúp chúng em hoàn thành báo cáo môn học này.

Cảm ơn tất cả các bạn trong lớp đã cho nhóm mình những ý kiến đóng góp và giúp đỡ trong quá trình làm báo cáo.

Tuy đã có những cố gắng nhất định, tìm hiểu và tiếp cận với đề tài nhưng do trình độ và thời gian hạn chế nên báo cáo này không thể tránh khỏi các thiếu sót. Rất mong nhận được những nhận xét góp ý và sửa sai của cô giáo hướng dẫn, các thầy cô, và các bạn để quyển báo cáo này trở nên hoàn thiện hơn.

Nhóm chúng em xin chân thành cảm ơn!

Hà Nội, ngày 4 tháng 1 năm 2020

Trưởng nhóm

Phùng Trọng Hiếu

Mục lục

Chương 1

Giới thiệu

Xã hội ngày càng hiện đại, các kỹ thuật công nghệ ngày càng phát triển, đi cùng với nó là các nghiên cứu phát triển không ngừng về lĩnh vực trí tuệ nhân tạo và học máy, cho ra đời các hệ thống máy móc thông minh ứng dụng rộng rãi trong hầu hết các lĩnh vực đời sống như máy truy tìm dữ liệu, chẩn đoán y khoa, phát hiện thẻ tín dụng giả, phân tích thị trường chứng khoán, phân loại chuỗi DNA, nhận dạng tiếng nói và chữ viết, ... đặc biệt là trong lĩnh vực điều khiển.

Chúng ta có rất nhiều loại thuật toán học như học có giám sát, học không có giám sát, học tăng cường, ... Mỗi loại thuật toán thích ứng với từng loại bài toán cụ thể. Trong báo cáo này, chúng ta sẽ nghiên cứu và tìm hiểu các vấn đề liên quan đến phương pháp học tăng cường (*Reinforcement Learning*). Đây là một thuật toán học có khả năng giải quyết được những bài toán thực tế khá phức tạp trong đó có sự tương tác giữa hệ thống và môi trường. Với những tình huống môi trường không chỉ đứng yên, cố định mà thay đổi phức tạp thì các phương pháp học truyền thống không còn đáp ứng được mà phải sử dụng phương pháp học tăng cường. Những bài toán với môi trường thay đổi trong thực tế là không nhỏ và ứng dụng nhiều trong các lĩnh lực quan trọng.

Môi trường thường được biểu diễn dưới dạng một quá trình quyết định Markov trạng thái hữu hạn (*Markov Decision Process - MDP*), và các thuật toán học tăng cường cho ngữ cảnh này có liên quan nhiều đến các kỹ thuật quy hoạch động. Các xác suất chuyển trạng thái và các xác suất thu lợi trong MDP thường là ngẫu nhiên nhưng lại là tĩnh trong quá trình của bài toán. MDP được biết đến sớm nhất là vào những năm 1950 (cf. Bellman 1957 [2]). Một cốt lõi của nghiên cứu về quá trình ra quyết định Markov là từ kết quả của cuốn sách của Ronald A.Howard (1960) [5]. Lý thuyết MDP được nghiên cứu bởi nhiều các nhà toán học, như Bertsekas (2005) [3] , White (1969) [11], và Puterman (1994) [6]. MDP cũng được nghiên cứu dưới dạng điều khiển tối ưu ngẫu nhiên, trong đó các phương pháp điều khiển tối ưu có liên quan chặt chẽ với việc học tăng cường.

Ví dụ đầu tiên mà trong đó việc học tăng cường được thảo luận bằng cách sử dụng MDP là Andreae's (1969b) [1] mô tả quan điểm về máy học. Witten and Corbin (1973) [13] đã thử nghiệm hệ thống học tăng cường, sau đó được phân tích bởi Witten (1977) [12] bằng cách sử dụng MDP. Mặc dù không đề cập rõ ràng về MDP, nhưng Werbos (1977) [10] đã đề xuất các phương pháp giải xấp xỉ cho các bài toán điều khiển tối ưu ngẫu nhiên, có liên quan tới các phương pháp học tăng cường hiện đại. Chúng được sử dụng trong rất nhiều các lĩnh vực khác nhau, bao gồm robot, điều khiển tự động, kinh tế và chế tạo.

Nội dung của báo cáo này được trình bày trong bốn chương.

- Chương 2: Quá trình Markov.

  Chương này trình bày một số định nghĩa về Xích Markov và Quá trình Markov.

- Chương 3: Quá trình quyết định Markov.

  Chương này trình này khái niệm quá trình quyết định Markov, bài toán Markov và các phần tử của bài toán Markov, phương trình tối ưu Bellman cho bài toán MDP.

- Chương 4: Thuật toán Q-Learning và Deep Q-Learning.

  Chương này giới thiệu về thuật toán Q-Learning và thuật toán Deep Q-Learning, sự hội tụ của thuật toán Q-Learning.

- Chương 5: Một số bài toán và ứng dụng cụ thể.

  Chương này trình bày hai ví dụ cụ thể về ứng dụng thuật toán Q-Learning và Deep Q-Learning trong giải quyết hai bài toán chiếc taxi thông minh và cân bằng con lắc ngược.

Chương 2

Quá trình Markov

Chương này trình bày định nghĩa, một số ví dụ về Xích Markov và Quá trình Markov. Kiến thức của chương được tổng hợp từ tài liệu [4].

2.1 Xích markov

Định nghĩa 2.1.1 Cho không gian trạng thái $E, \{S_t\}_{t\geq 0}$ được gọi là một xích Markov nếu nó thỏa mãn tính không nhớ:

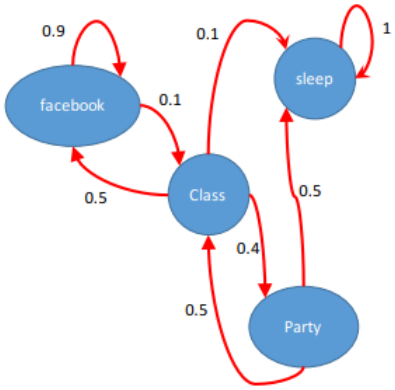
$$P(S_{t+1} = s_{t+1} | S_t = s_t, S_{t-1} = s_{t-1}, ..., S_0 = s_0) = P(S_{t+1} = s_{t+1} | S_t = s_t), \qquad (2.1)$$

với $t \in \mathbb{N}$ và $s_i \in E, i = 0, 1, ..., t+1$.

Ví dụ 2.1.2 *Xích Markov các trạng thái của một sinh viên trong lớp với xác suất chuyển được minh họa như Hình 2.1.*

- *Tập không gian trạng thái $S = \{FB, C, Pt, Sl\}$ tương ứng với các hoạt động Facebook, Class, Party và Sleep của sinh viên.*
- *Ma trận xác suất chuyển:*

$$\begin{bmatrix} 0.9 & 0.1 & 0 & 0 \\ 0.5 & 0 & 0.4 & 0.1 \\ 0 & 0.5 & 0 & 0.5 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$



Hình 2.1: Sơ đồ chuyển trạng thái của sinh viên.

2.2 Quá trình Markov

Định nghĩa 2.2.1 $\{S_t\}_{t\geq 0}$ với không gian trạng thái hữu hạn E được gọi là quá trình Markov nếu thỏa mãn :

$$P(S_{t+s} = j|S_u; u \leq t) = P(S_{t+s} = j|S_t). \tag{2.2}$$

Nhận xét 2.2.2 Nếu $Pr(S_{t+s} = j|S_t = i) = Pr(S_s = j|S_0 = i)$ thì được gọi là quá trình Markov dừng.

Chương 3

Quá trình quyết định Markov - MDP

Quá trình quyết định Markov (MDP) cung cấp một nền tảng toán học cho việc mô hình hóa ra quyết định trong các tình huống mà kết quả có yếu tố ngẫu nhiên và/hoặc phụ thuộc vào sự điều khiển của một người ra quyết định. MDP được sử dụng rất nhiều các lĩnh vực khác nhau, bao gồm robot, điều khiển tự động, kinh tế, chế tạo, ... Trong phần này, chúng ta sẽ trình bày về quá trình quyết định Markov trong đó tập trung vào các khái niệm của quá trình Markov có số bước vô hạn và hữu hạn. Kiến thức trong phần này được tổng hợp từ các tài liệu [2], [7] và [6].

3.1 Khái niệm

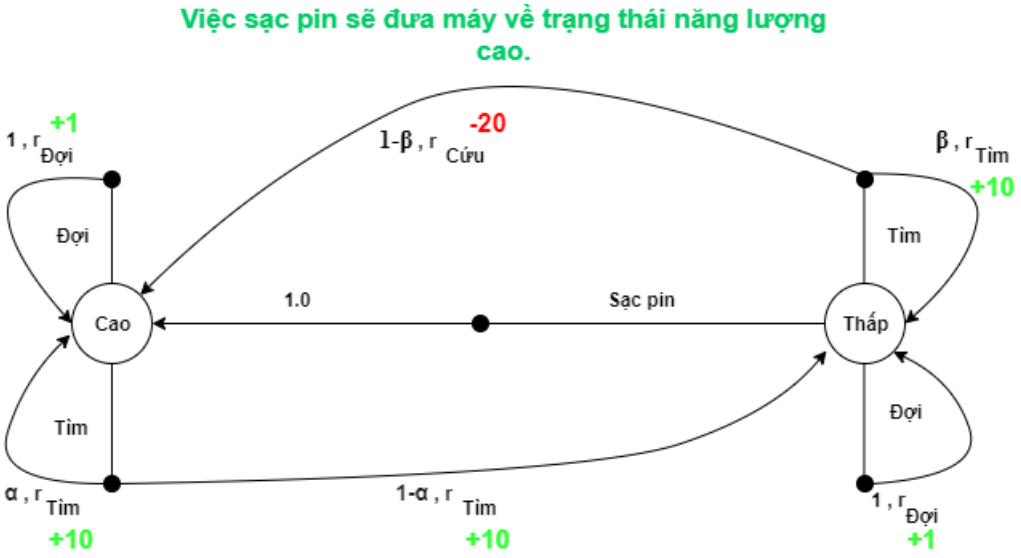
Một quá trình quyết định Markov (*Markov Decision Process*) là một quá trình phần thưởng Markov (*Markov Reward Process*) với các quyết định. Trong đó, tất cả các trạng thái đều thỏa mãn tính Markov. Cụ thể:

Định nghĩa 3.1.1 (Quá trình quyết định Markov) Một quá trình quyết định Markov là một bộ năm thành phần $(S, A, P(.|.,.), R(.,.), \gamma)$, trong đó:

- S là một tập hữu hạn các trạng thái, kí hiệu S_t là trạng thái tại thời điểm t;
- A là một tập hữu hạn các hành động, kí hiệu A_t là hành động tại thời điểm t ($A(s)$ là tập hữu hạn các hành động có sẵn từ trạng thái s với $s \in S$);
- $Pr_a(s,s') = Pr(S_{t+1} = s' | S_t = s, A_t = a)$ là xác suất mà hành động a tại trạng thái s ở thời điểm t chuyển sang trạng thái s' tại thời điểm $t+1$. Xác suất này thỏa mãn tính Markov, nghĩa là:

$$Pr(S_{t+1} = j | S_0, A_0, ..., S_t, A_t) = Pr(S_{t+1} = j | S_t, A_t);$$

- $R_a(s,s')$ là phần thưởng nhận được khi chọn hành động s để chuyển trạng thái từ s sang s';
- $\gamma \in [0,1]$ là hệ số suy giảm, đại diện cho sự khác biệt giữa các phần thưởng trong tương lai và phần thưởng hiện tại.



Hình 3.1: Quá trình chuyển trạng thái của máy dọn rác tự động.

Ví dụ 3.1.2 : Máy bán hàng tự động

- Trạng thái: cấu hình các khe.
- Hành động: thời gian dừng lại.
- Mục tiêu: kiếm được nhiều tiền.
- Bài toán: tìm $\pi : S \to A$ sao cho R lớn nhất.

Định nghĩa 3.1.3 (Chiến lược quyết định). Một *chiến lược* π là một phân phối các hành động tại mỗi trạng thái:

$$\pi(a|s) = Pr[A_t = a|S_t = s].$$

Một số kiểu chiến lược như sau:

- $\pi = (\pi^0, ..., \pi^{T-1})$ được gọi là *một chiến lược Markov* nếu tại mỗi thời điểm n, quyết định π^n không phụ thuộc vào quá khứ h_{n-1}.
  Ví dụ:

$$\pi^n_{h_{n-1},i_n}(a) = \pi^n_{h'_{n-1},i_n}(a),\ a \in A(i_n),\ i_n \in S\ \forall\ h_{n-1},\ h'_{n-1}.$$

- π được gọi là một chiến lược dừng nếu mọi quy tắc quyết định đều bằng nhau :

$$\pi^n = \pi^0 \text{ với } n = 0, 1, ..., T-1.$$

Trong quá trình quyết định Markov, mỗi chiến lược xác định đầy đủ hành động của một tác nhân. Ngoài ra, các chiến lược này chỉ phụ thuộc vào trạng thái của xích Markov tại thời điểm hiện tại (không phụ thuộc vào quá khứ).

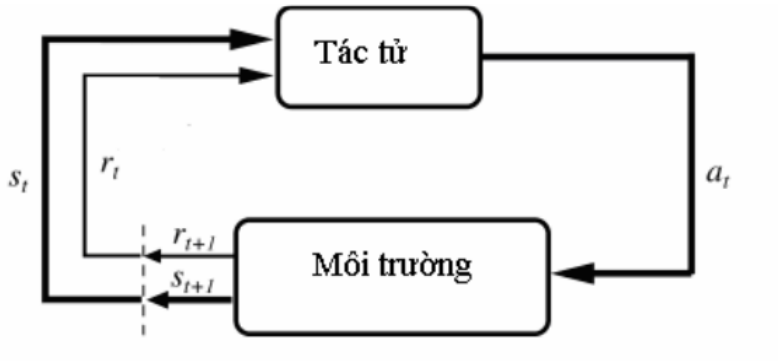
3.2 Bài toán quyết định Markov

3.2.1 Phát biểu bài toán

Bài toán quyết định Markov là bài toán học từ các tác động để đạt được mục đích. Người học và người ra quyết định được gọi là tác tử. Tất cả những gì mà chúng tương tác với, bao gồm mọi thứ bên ngoài tác tử được gọi là môi trường. Các tác động thực hiện một cách liên tục, tác tử lựa chọn hành động, môi trường đáp ứng lại các hành động đó và chuyển từ trạng thái hiện thời sang trạng thái mới. Môi trường cũng đem lại các mục tiêu, các giá trị hằng số mà tác tử cố gắng cực đại hóa qua thời gian. Một đặc tả hoàn thiện về môi trường được coi là một "nhiệm vụ", một thực thể của bài toán quyết định Markov.
Tóm lại, bài toán quyết định Markov liên quan đến lớp bài toán trong đó một tác tử rút ra kết luận trong khi phân tích một chuỗi các hành động của nó cùng với tín hiệu vô hướng được đưa ra bởi môi trường.
Trong khái niệm chung này, có thể thấy hai đặc tính quan trọng:

- Tác tử tương tác với môi trường và cặp "tác tử + môi trường" tạo thành một hệ thống động.
- Tín hiệu tăng cường, được nhận biết dựa vào mục tiêu, cho phép tác tử thay đổi hành vi của nó.

Lược đồ tương tác tác tử-môi trường như sau:



Hình 3.2: Mô hình tương tác giữa tác tử và môi trường.

Trong lược đồ trên, tác tử và môi trường tác động lẫn nhau tại mỗi bước trong chuỗi các bước thời gian rời rạc, $t = 0, 1, 2, 3, \ldots$. Tại mỗi bước thời gian t, tác tử nhận một số biểu diễn về trạng thái của môi trường, $s_t \in S$, với S là tập các trạng thái có thể, và trên đó lựa chọn một hành động $a_t \in A(s_t)$, với $A(s_t)$ là tập các hành động trong trạng thái s_t. Mỗi bước thời gian tiếp theo, tác tử nhận một giá trị tăng cường $r_{t+1} \in R$ và tự nó tìm ra một trạng thái mới S_{t+1}. Tại mỗi bước tác tử thực hiện ánh xạ từ các trạng thái đến các hành động có thể lựa chọn. Phép ánh xạ này được gọi là chiến lược của tác tử, kí hiệu là π_t với $\pi_t(s,a)$ là xác suất thực hiện hành động $a_t = a$ khi $s_t = s$.

3.2.2 Các phần tử của bài toán quyết định Markov

Dựa vào tác tử và môi trường, chúng ta có thể định nghĩa các phần tử con của một bài toán quyết định Markov: chiến lược (*policy*), hàm phản hồi (*reward function*), hàm giá trị (*value function*), và không bắt buộc, một mô hình về môi trường.

Chiến lược

Chiến lược định nghĩa cách thức tác tử học từ hành động tại thời điểm đưa ra. Chiến lược là một ánh xạ từ tập các trạng thái của môi trường đến tập các hành động được thực hiện khi môi trường ở trong các trạng thái đó. Nó tương ứng với tập các luật nhân quả trong lĩnh vực tâm lý học. Trong một số trường hợp, chiến lược có thể là một hàm đơn giản hoặc một bảng tra cứu, trong những trường hợp khác, nó có thể liên quan đến các tính toán mở rộng, ví dụ như một tiến trình tìm kiếm. Chiến lược là nhân của một tác tử với nhận thức rằng một mình nó đủ quyết định hành động.

Hàm phản hồi

Mục đích của tác tử là cực đại hóa các mục tiêu được tích lũy trong tương lai. Hàm phản hồi $R(t)$ được biểu diễn dưới dạng hàm số đối với các mục tiêu. Trong các bài toán quyết định Markov, hàm phản hồi sử dụng biểu thức dạng tổng. Các nhà nghiên cứu đã tìm ra hai biểu diễn thường được sử dụng của hàm phản hồi:

- *Trong các bài toán số bước hữu hạn*

  Với những bài toán này ta có một số hữu hạn các bước trong tương lai. Sẽ tồn tại một trạng thái kết thúc và một chuỗi các hành động giữa trạng thái đầu tiên và trạng thái kết thúc được gọi là một giai đoạn. Ta có:

  $$R(t) = r_t + r_{t+1} + ... + r_{t+K-1}$$

  Trong đó K là số các bước trước trạng thái kết thúc.

- *Trong các bài toán số bước vô hạn*

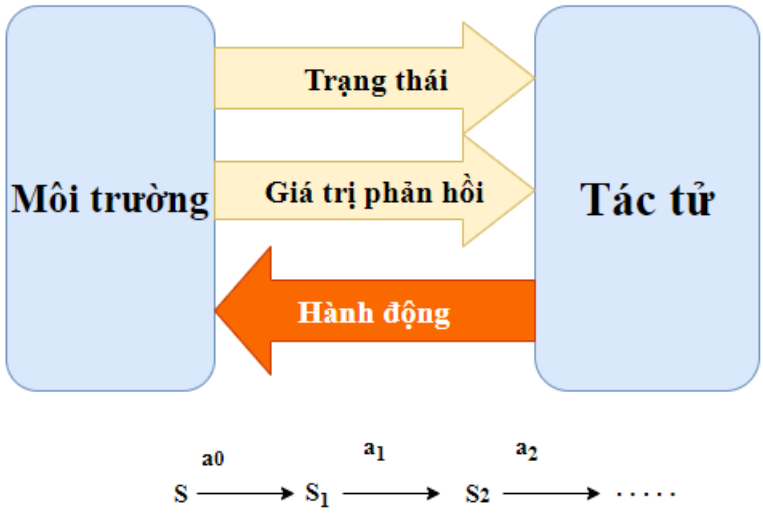
  Với những bài toán này ta có chuỗi các hành động là vô hạn. Một hệ số suy giảm $\gamma, 0 \leq \gamma \leq 1$ được đưa ra và hàm phản hồi được biểu diễn dưới dạng tổng của các giá trị mục tiêu giảm dần:

  $$R(t) = \sum_{k=0}^{\infty} \gamma^k r_{t+k}.$$

  Hệ số γ cho phép xác định mức độ ảnh hưởng của những bước chuyển trạng thái tiếp theo đến giá trị phản hồi tại thời điểm đang xét. Giá trị của γ cho phép điều chỉnh giai đoạn tác tử lấy các hàm tăng cường. Nếu $\gamma = 0$, thì tác tử chỉ xem xét mục tiêu gần nhất, giá trị γ càng gần với 1 thì tác tử sẽ quan tâm đến các mục tiêu xa hơn trong tương lai. Như vậy, thực chất bài toán quyết định Markov trong trường hợp này chính là việc lựa chọn các hành động để làm cực đại biểu thức R:

  $$R = r_0 + \gamma r_1 + \gamma^2 r_2 + ... \text{ với } 0 < \gamma < 1.$$

  Như trong hình vẽ minh họa sau:



Hình 3.3: Mô hình tương tác giữa tác tử và môi trường trong bài toán có số bước vô hạn.

Hàm giá trị

Trong mọi trạng thái s_t, một tác tử lựa chọn một hành động dựa theo một chiến lược điều khiển, $\pi : a_t = \pi(s_t)$. Hàm giá trị tại một trạng thái của hệ thống được tính bằng kỳ vọng toán học của hàm phản hồi theo thời gian. Hàm giá trị là hàm của trạng thái và xác định mức độ thích hợp của chiến lược điều khiển π đối với tác tử khi hệ thống đang ở trạng thái s. Hàm giá trị của trạng thái s trong chiến lược π được tính như sau:

$$V^{\pi}(s) = E_{\pi}\{R_t | s_t = s\}.$$

Bài toán tối ưu bao gồm việc xác định chiến lược điều khiển π^* sao cho hàm giá trị của trạng thái hệ thống đạt cực đại sau một số vô hạn hoặc hữu hạn các bước:

$$\pi^* = \{\pi_0(s_0), \pi_1(s_1), ..., \pi_{N-1}(s_{N-1})\}.$$

Đối với bài toán có số bước vô hạn ta có hàm giá trị trạng thái:

$$V^{\pi}(s) = E_{\pi}\{R_t = \sum_{k=0}^{\infty} \gamma^k r_{t+k+1} | s_t = s\}.$$

Sử dụng các phép biến đổi:

$$\begin{aligned}
V^{\pi}(s) &= E_{\pi}\{R_t|s_t = s\} \\
&= E_{\pi}\{\sum_{k=0}^{\infty}\gamma^k r_{t+k+1}|s_t = s\} \\
&= E_{\pi}\{r_{t+1} + \gamma\sum_{k=0}^{\infty}\gamma^k r_{t+k+2}|s_t = s\} \\
&= \sum_a \pi(s,a)\sum_{s'} P^a_{ss'}[R^a_{ss'} + \gamma E_{\pi}\{\sum_{k=0}^{\infty}\gamma^k r_{t+k+2}|s_{t+1} = s'\}] \\
&= \sum_a \pi(s,a)\sum_{s'} P^a_{ss'}[R^a_{ss'} + \gamma V^{\pi}(s')].
\end{aligned}$$

Như vậy, hàm $V^{\pi}(s)$ có thể được viết lại một cách đệ qui như sau:

$$V^{\pi}(s) = E_{\pi}r_{t+1} + \gamma V^{\pi}(S_{t+1})|S_t = s,$$

hay

$$V^{\pi}(s) = R(s,a) + \gamma\sum_{s'\in S} P^a_{ss'}V^{\pi}(s'). \tag{3.1}$$

Với $P^a_{ss'}$ là xác suất để chuyển từ trạng thái s sang s' khi áp dụng hành động a. Chúng ta có thể tính toán hàm $V^{\pi}(s)$ ngoại tuyến nếu biết trạng thái bắt đầu và xác suất mọi phép chuyển đổi theo mô hình. Vấn đề đặt ra là sau đó giải quyết hệ thống các phương trình tuyến tính trong công thức 3.1. Chúng ta biết rằng tồn tại một chiến lược tối ưu, kí hiệu π^*, được định nghĩa như sau :

$$\begin{aligned}
V^{\pi^*}(s) &\geq V^{\pi}(s) \\
\pi^* &= argmax_{\pi}V^{\pi}(s),
\end{aligned}$$

để đơn giản chúng ta viết $V^* = V^{\pi^*}$. Hàm giá trị tối ưu của một trạng thái tương ứng với chiến lược tối ưu là:

$$V^{\pi}(s) = max_{\pi}V^{\pi}(s),$$

đây là phương trình tối ưu Bellman (hoặc phương trình của quy hoạch động) ta sẽ nói chi tiết hơn ở phần tiếp theo.

Tóm lại V^{π} *là hàm giá trị trạng thái cho chiến lược* π. Giá trị của trạng thái kết thúc thường bằng 0. Tương tự, định nghĩa $Q^{\pi}(s,a)$ là giá trị của việc thực hiện hành động a trong trạng thái s dưới chiến lược điều khiển π, được tính bằng kỳ vọng toán học của hàm phản hồi bắt đầu từ trạng thái s, thực hiện hành động a trong chiến lược π:

$$.Q^{\pi}(s,a) = E_{\pi}\{R_t|s_t = s, a_t = a\} = E_{\pi}\Big\{\sum_{k=0}^{\infty}\gamma^k r_{t+k+1}|s_t = s, a_t = a\Big\},$$

Q^π được gọi là hàm giá trị hành động cho chiến lược π. Và các hàm giá trị V^π, Q^π có thể được ước lượng từ kinh nghiệm.

3.3 Phương trình tối ưu Bellman cho bài toán MDP

Từ trạng thái s, có thể đưa ra nhiều hành động khác nhau và mỗi chiến lược xác định một phân phối xác suất của hành động đó, do đó, sử dụng phương trình tối ưu Bellman để đưa ra quyết định cho bài toán này.
Phương trình Bellman cho hàm giá trị trạng thái:

$$\begin{aligned} V^\pi(s) &= R_\pi(s) + \gamma \sum_{s' \in S} P_\pi(s,s') V^\pi(s') \\ &= \sum_{a \in A} \pi(a|s) \left(R_a(s) + \gamma \sum_{s' \in S} P_\pi(s,s') V^\pi(s') \right). \end{aligned}$$

Phương trình Bellman cho hàm giá trị hành động:

$$Q^\pi(s,a) = R_a(s) + \gamma \sum_{s' \in S} P_a(s,s') \sum_{a' \in A} \pi(a'|s') Q^\pi(s',a').$$

Định nghĩa 3.3.1 Hàm giá trị trạng thái tối ưu $V^*(s)$ là hàm trả về trạng thái tối ưu trên tất cả các chiến lược:

$$V^*(s) \ = \ \max_\pi V^\pi(s).$$

Định nghĩa 3.3.2 Hàm giá trị hành động tối ưu $Q^*(s,a)$ là hàm trả về các hành động tối ưu trên tất cả các chiến lược:

$$Q^*(s,a) \ = \ \max_\pi Q^\pi(s,a).$$

Định nghĩa 3.3.3 (Chiến lược tối ưu) π được gọi là chiến lược tối ưu nếu $V^\pi(s) \geq V^{\pi'}(s), \forall s$.
Xác định một chiến lược tối ưu: Một chiến lược tối ưu có thể được xác định bằng cách tìm hàm cực đại $Q^*(s,a)$:

$$\pi^*(a|s) = \begin{cases} 1 & \text{nếu } a = argmax_{a \in A}\ Q^*(s,a) \\ 0 & \text{ngược lại} \end{cases}$$

Phương trình tối ưu Bellman cho $V^*(s)$:

$$V^*(s) = \max_a \ R_a(s) + \gamma \sum_{s' \in S} P_a(s,s') V^*(s').$$

Phương trình tối ưu Bellman cho $Q^*(a,s)$:

$$Q^*(s,a) = R_a(s) + \gamma \sum_{s' \in S} P_a(s,s') \max_{a'} Q^*(s',a').$$

Phương trình tối ưu Bellman là một phương trình phi tuyến có thể giải bằng một số phương pháp: lặp giá trị và lặp chính sách.

Chương 4

Thuật toán Q-Learning và Deep Q-Learning

4.1 Q-Learning

4.1.1 Giới thiệu

Những thuật toán học tăng cường thông thường gồm có quy hoạch động (Dynamic Programming), Monte-Carlo và phương pháp TD (Temporal-Difference). Tuy nhiên các phương pháp quy hoạch động và Monte-Carlo không hiệu quả do đòi hỏi bộ nhớ quá lớn, hoặc mô hình phải xác định hay khó hội tụ nên ít khi cho ra kết quả tối ưu. Phương pháp TD là sự kết hợp của hai phương pháp quy hoạch động, Monte-Carlo và nó còn cho phép giải quyết được nhiều bài toán thực tế bởi vì phương pháp này không đòi hỏi môi trường xác định và có khả năng hội tụ cao. Một biến thể của phương pháp TD được gọi là Q-learning. Nó là một phương pháp học kiểu TD theo hướng off-policy, rất hiệu quả trong việc giải quyết các bài toán tìm đường. Quá trình học tăng cường có thể được thực hiện theo hai cách: off-policy và on-policy. On-policy sử dụng một chiến lược chọn hành động để thực hiện các bước hành động để tối ưu hóa chính chiến lược chọn hành động đó. Phương pháp off-policy sử dụng một chiến lược chọn hành động để thực hiện các hành động nhưng với mục đích là để tối ưu hóa một chiến lược chọn hành động khác.

4.1.2 Thuật toán Q-Learning

Cho một chiến lược (policy) π, ta viết lại công thức (trình bày ở chương 3) về Q-value (hay là giá trị hành động) như sau:

$$Q^{\pi}(s,a) = R(s,a) + \gamma \sum_{s' \in \S} P^{a}_{ss'} V^{\pi}(s'). \tag{4.1}$$

Trong công thức 4.1, $Q^{\pi}(s,a)$ là Q-value khi thực hiện hành động a tại trạng thái s theo chiến lược π; $R(s,a)$ là phần thưởng nhận được; $s^{'}$ là trạng thái kế tiếp. γ là hệ số giảm (discount rate), đảm bảo "gần" đích Q-value càng lớn.
Nói các khác, Q-value là hàm giá trị hành động mong đợi cho việc thực hiện hành động a ở trạng thái s và tuân theo chính sách π sau đó. Mục đích của Q-learning là ước lượng Q-values cho một chiến lược tối ưu. Để tiện lợi, ta định nghĩa như sau $Q^*(s,a) \equiv Q^{\pi^*}(s,a), \forall s,a$. Thật đơn giản để chỉ ra rằng $V^*(s) = max_a Q^*(s,a)$ và nếu a^* là một hành động mà tại đó mức tối đa đạt được, sau đó một chiến lược tối ưu có thể được hình thành như $\pi^*(s) \equiv a^*$. Ở đây, lợi ích của những giá trị Q-value là nếu một tác tử (agent) có thể học chúng, nó có thể dễ dàng quyết định đâu là tối ưu để hành động. Mặc dù có thể có nhiều hơn một chiến lược tối ưu hay a^*, những giá trị Q^* là duy nhất.

Trong Q-learning, kinh nghiệm của tác tử (agent) bao gồm một chuỗi tuần tự các quá trình riêng biệt (gọi là Episodes).
Trong episode thứ n, tác tử sẽ:

- quan sát trạng thái hiện tại của nó s_n,
- lựa chọn và thực hiện một hành động a_n,
- quan sát trạng thái tiếp theo s_{n+1},
- nhận được ngay phần thưởng r_n, và
- điều chỉnh các giá trị Q_{n-1} của nó bằng cách sử dụng hệ số học α_n, theo:

$$Q_n(s,a) = \begin{cases} (1-\alpha_n)Q_{n-1}(s,a) + \alpha_n[r_n + \gamma V_{n-1}(s_{n+1})] & \text{nếu } s = s_n \text{ và } a = a_n, \\ Q_{n-1}(s,a) & \text{trường hợp khác} \end{cases} \tag{4.2}$$

Trong đó :

$$V_{n-1}(s) \equiv max_b Q_{n-1}(s,b) \tag{4.3}$$

là tốt nhất mà tác tử có thể làm được từ trạng thái s. Dĩ nhiên, ở những giai đoạn đầu việc học thì Q-value có thể không phản ánh đúng chiến lược mà chúng đã được biết (việc tối đa hóa các hành động ở phương trình 4.3). Hiển nhiên là các giá trị Q-value ban đầu, $Q_0(s,a)$, cho tất cả các trạng thái và hành động đã được giả thiết. Ngoài ra ta có thêm giải thiết một bảng tra cứu (look-up table) biểu diễn cho $Q_n(s,a)$. Theo [8] cho thấy Q-learning có thể không hội tụ chính xác cho các biểu diễn khác.

Phân tích thuật toán Q-learning, ta có các bước như sau:

- 1. Khởi tạo bảng Q với các số 0 và giá trị Q thành các hằng số tùy ý.

- 2. Khám phá các hành động: đối với mỗi thay đổi về trạng thái, chọn bất kỳ một hành động (a) nào trong số tất cả các hành động có thể có cho trạng thái hiện tại (S).
- 3. Đi đến trạng thái tiếp theo (S') là kết quả của hành động (a).
- 4. Đối với tất cả các hành động có thể từ trạng thái (S,), hãy chọn một hành động có giá trị Q cao nhất.
- 5. Cập nhật giá trị bảng Q bằng phương trình.
- 6. Đặt trạng thái tiếp theo làm trạng thái hiện tại.
- 7. Nếu trạng thái cuối đạt được, sau đó kết thúc và lặp lại quá trình.

4.1.3 Sự hội tụ của thuật toán

Điều kiện quan trọng nhất trong định lý hội tụ được đưa ra dưới đây là chuỗi các quá trình (Episodes) hình thành nền tảng học tập phải bao gồm vô hạn quá trình cho mỗi trạng thái và hành động bắt đầu. Đây có thể được coi là một điều kiện mạnh về cách lựa chọn các trạng thái và hành động, tuy nhiên, trong các điều kiện ngẫu nhiên của định lý, không có phương pháp nào có thể được đảm bảo để tìm ra một chiến lược tối ưu trong các điều kiện yếu hơn. Tuy nhiên, xin lưu ý rằng các quá trình không cần phải tạo thành một chuỗi liên tục, đó là y của một quá trình không cần phải là x của quá trình tiếp theo.

Định lý dưới đây định nghĩa một tập các điều kiện theo đó $Q_n(s,a) \to Q^*(s,a)$ khi $n \to \infty$. Định nghĩa $n^i(s,a)$ là chỉ số của lần thứ i mà hành động a được thử ở trạng thái s. Định lý hội tụ thuật toán Q-learning:

Định lý 4.1.1 *Cho một miền phần thưởng bị chặn $r_n \leq R$, tỉ lệ học $0 \leq \alpha_n < 1$, và*

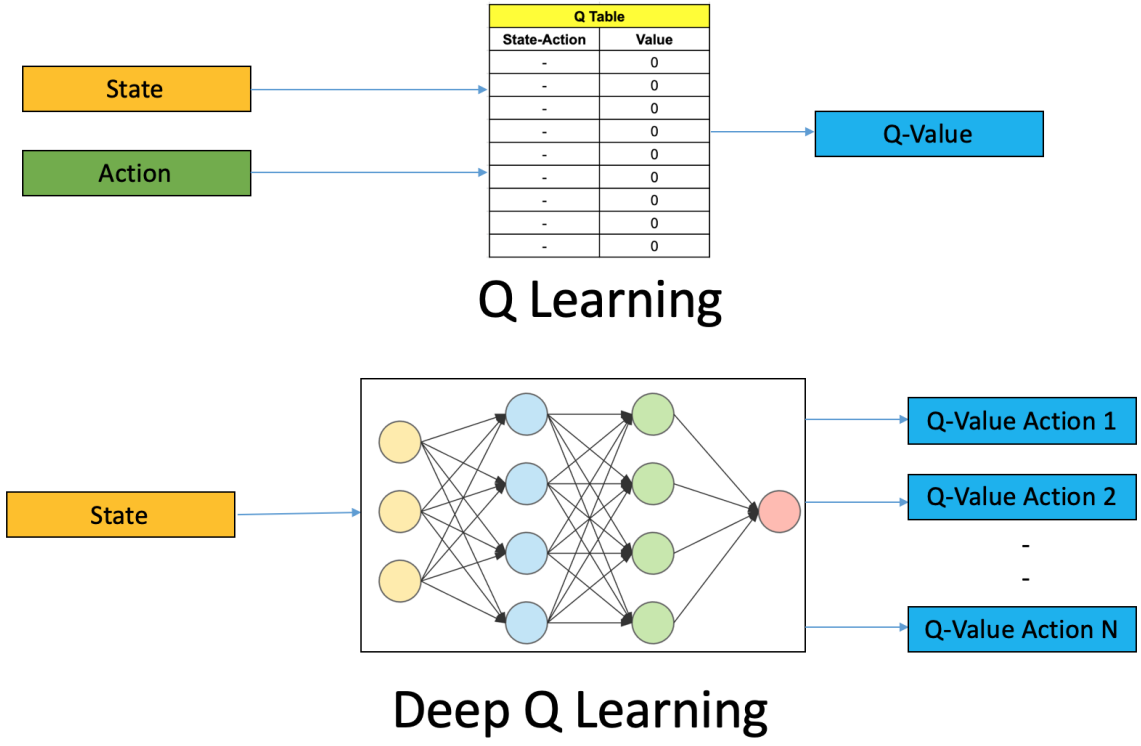
$$\sum_{i=1}^{\infty} \alpha_{n^i(s,a)} = \infty, \sum_{i=1}^{\infty} [\alpha_{n^i(s,a)}]^2 < \infty, \forall s, a, \tag{4.4}$$

thì $Q_n(s,a) \to Q^*(s,a)$ khi $n \to \infty, \forall x, a$, với xác suất bằng 1.

Chứng minh tính đúng đắn của định lý 4.1.1 xem chi tiết trong [9].

4.2 Deep Q-Learning

Mục đích của bài toán là chọn ra hành động (action) thích hợp cho một trạng thái nào đó (state) nào đó. Hay nói cách khác, với state là đầu vào và cần đầu ra là một hành động. Công việc đó sẽ được thực hiện qua một mạng nơ-ron (Neural Network-NN). Những gì ta cần làm chỉ là bỏ đi bảng tra cứu lookup table $Q(s,a)$ và thay thế bằng một mạng nơ-ron đơn giản. Điều đó được mô tả trong Hình 4.1.



Hình 4.1: So sánh về cấu trúc giữa Q-learning và Deep Q-learning.

Hàm mất mát

Mục đích của ta là bắt mạng học được cách ước lượng Q-Value cho các hành động một cách chính xác nên đương nhiên hàm mất mát phải tính được sai số giữa Q-value thực tế và dự đoán. Hàm mất mát định nghĩa dưới dạng đầy đủ như sau:

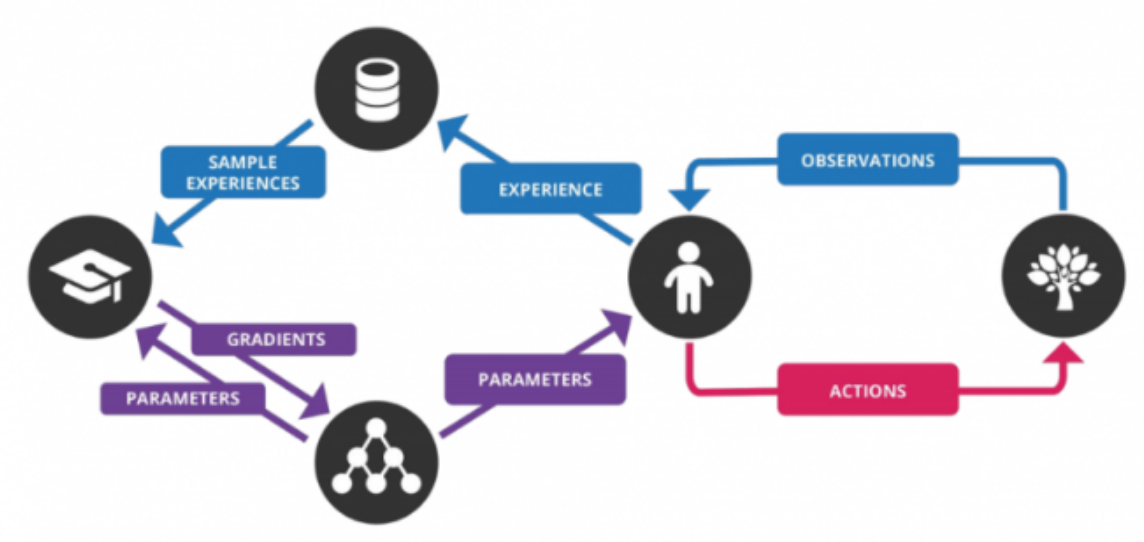
$$Loss = (r + \gamma max_{a'} Q(s', a'; \theta') - Q(s, a; \theta))^2 \tag{4.5}$$

Kinh nghiệm chơi lại (Experience replay)

Ở phần trên ta đã định nghĩa một mạng nơ-ron lấy input là state hiện tại và output các Q-value. Thế nhưng nếu mạng nơ-ron cứ liên tục bị đẩy vào từng state một sẽ rất dễ bị overfitting vì các states liên tục thường giống nhau hoặc có tính tuyến tính (ví dụ: liên tục đi thẳng/sang trái/phải). Kỹ thuật Experience Replay được sử dụng để loại bỏ vấn đề này. Thay vì mỗi state mạng update một lần, ta lưu lại các states vào bộ nhớ (memory). Sau đó thực hiện sampling thành các batch nhỏ đưa vào mạng nơ-ron học. Việc này giúp đa dạng hóa input và tránh mạng nơ-ron bị overfitting.

Mô hình

Mô hình đầy đủ của Deep Q-Learning được mô tả trong Hình 4.2



Hình 4.2: Mô hình Deep Q-Learning.

Tóm lại

Deep Q-Learning thực hiện các bước sau:

- 1. Enviroment đưa vào mạng một state s; đầu ra là các Q-value của các actions tương ứng.
- 2. Agent chọn action bằng một Policy và thực hiện action a đó.
- 3. Environment trả lại state s' cùng với reward r tương ứng và lưu lại experience s, a, r, s'.
- 4. Thực hiện sample các experience thành một vài batches và tiến hành luyện mạng.
- 5. Lặp lại đến khi kết thúc M episodes.

Chương 5

Một số bài toán ứng dụng

Trong chương này, ta sẽ cùng nhau giải quyết hai bài toán học tăng cường cơ bản để minh họa cho việc sử dụng thuật toán Q-Learning đã được trình bày ở chương trước. Bài toán đầu tiên sẽ là bài Chiếc taxi thông minh (Smart Taxi); trong bài toán này, ta sẽ huấn luyện một chiếc taxi sao cho nó có thể đón và trả khách tại đúng vị trí và thực hiện việc này một cách "thông minh" nhất có thể. Bài toán thứ hai sẽ giải quyết việc điều khiển chiếc xe đẩy để giữ cho con lắc được gắn trên xe luôn ở trạng thái cân bằng; bài toán kinh điển này còn được biết đến với cái tên Bài toán cân bằng con lắc ngược (CartPole).

5.1 Bài toán chiếc taxi thông minh

5.1.1 Bài toán

Ta có một chiếc taxi được trang bị các cảm biến, trí tuệ nhân tạo, ... để có thể tự vận hành trong mọi điều kiện giao thông và thời tiết. Nhiệm vụ của chiếc xe này là đón và trả khách tại những vị trí nhất định. Ngoài ra, việc vận chuyển hành khách cần phải thỏa mãn những tiêu chí sau:

- Phải trả khách tại đúng vị trí được chỉ định
- Tiết kiệm thời gian cho hành khách một cách tối đa
- Đảm bảo hành khách được an toàn và phải tuân thủ tất cả các luật giao thông được đưa ra

5.1.2 Mô hình hóa bài toán

Trước khi có thể sử dụng các kỹ thuật học tăng cường để huấn luyện cho chiếc taxi (agent của chúng ta) thực hiện công việc đưa đón khách một cách tự động, ta cần phải quan tâm đến một vài khía cạnh về việc mô hình hóa bài toán. Ta cần phải biết phần thưởng (rewards), không gian trạng thái (state space) của chiếc taxi, và các hành động (actions) mà chiếc taxi có thể thực hiện tại mỗi trạng thái.

Phần thưởng

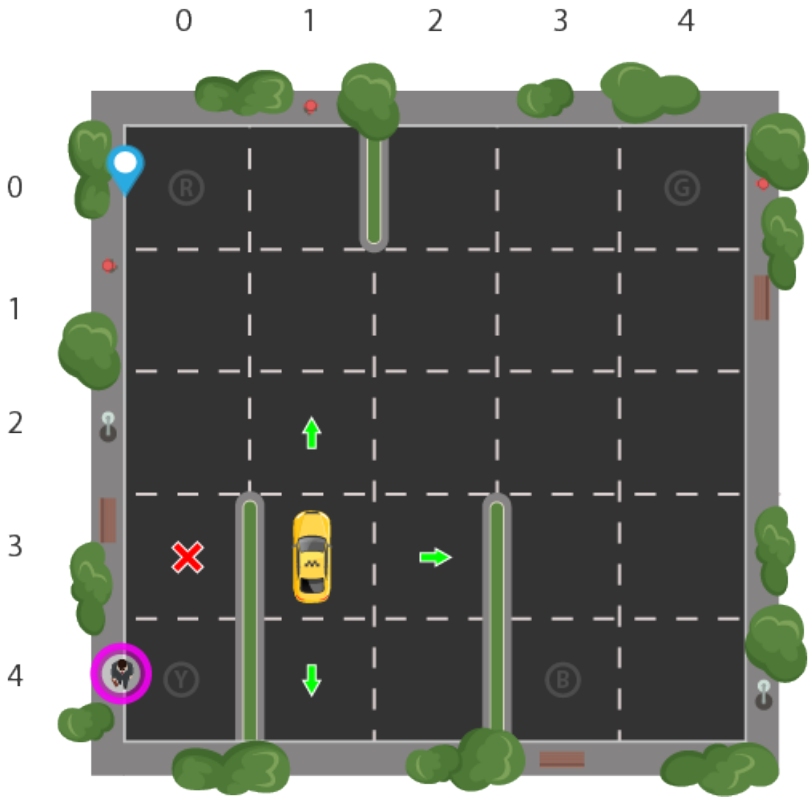
Vì chiếc taxi sẽ được huấn luyện bằng cách thử và sai khi tương tác với môi trường, ta cần phải định nghĩa phần thưởng và/hoặc hình phạt cho nó:

- Chiếc taxi (agent) sẽ nhận được một phần thưởng lớn (+20 điểm) khi trả khách thành công (trả đúng vị trí được đưa ra)
- Chiếc taxi sẽ bị phạt nặng nếu nó trả khách sai vị trí (-10 điểm)
- Chiếc taxi sẽ bị phạt "nhẹ" (slight negative reward) trong suốt chuyến hành trình đi đến vị trí trả khách (-1 điểm/bước). Hình phạt ở đây không được lớn vì ta không muốn việc chiếc taxi cố gắng "lao" đến đích một cách nhanh nhất có thể mà vi phạm luật giao thông hay gây nguy hiểm cho hành khách.

Không gian trạng thái

Không gian trạng thái là tập chứa tất cả những tình huống mà chiếc taxi của chúng ta có thể gặp phải. Đây là nơi chứa những thông tin vô cùng cần thiết cho chiếc taxi để giúp nó có thể đưa ra những hành động "đúng" tương ứng với trạng thái mà nó đang ở.

Giả sử, ta có một bãi tập cho chiếc taxi của chúng ta, như được minh họa trong Hình 5.1; ở đây, ta sẽ dạy chiếc taxi vận chuyển hành khách đến các vị trí (R, G, Y, B) trên bãi tập.



Hình 5.1: Bãi tập.

Để đơn giản hóa bài toán, ta có một số giả định như sau:

- Chiếc taxi là phương tiện duy nhất có trên bãi tập
- Khu vực huấn luyện có thể được chia thành một lưới 5×5, cho ta tổng cộng 25 vị trí mà chiếc taxi có thể đỗ. Ví dụ, như ta có thể thấy trên Hình 5.1, chiếc taxi đang nằm tại vị trí có tọa độ (3, 1); ngoài ra 4 vị trí R, G, Y, B có tọa độ lần lượt là (0,0), (0,4), (4,0), (4,3); vị hành khách đáng kính đang đứng tại vị trí Y và có mong muốn di chuyển đến vị trí R trên bãi tập.

Vậy, ta có tổng cộng $5 \times 5 = 25$ vị trí mà chiếc taxi có thể xuất hiện, 4 đích đến, và 5 vị trí của hành khách (4 vị trí tại R, G, Y, B và 1 vị trí là ở trên chiếc taxi). Tổng số trạng thái có thể có của môi trường sẽ là $5 \times 5 \times 5 \times 4 = 500$ trạng thái.

Không gian hành động

Tại mỗi thời điểm, chiếc taxi (agent của bài toán) sẽ nằm ở 1 trong tổng 500 trạng thái, và nó sẽ thực hiện một hành động tương ứng với trạng thái hiện có. Hành động ở đây có thể là đón/trả khách, và di chuyển quanh bãi tập.

Không gian hành động của ta sẽ gồm:

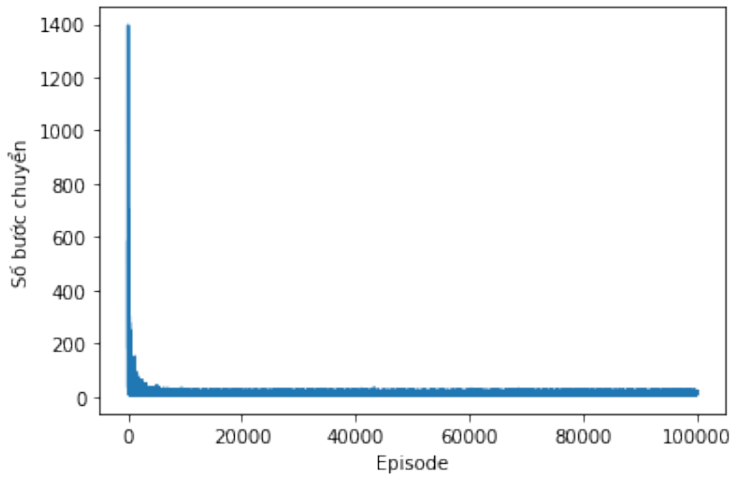
- Đi lên
- Đi xuống
- Đi sang trái
- Đi sang phải
- Đón khách
- Trả khách

Để ý rằng, tại một số trạng thái ta không thể thực hiện một vài hành động nhất định. Ví dụ như khi chiếc taxi ở vị trí mép tường bên trái, nó không thể thực hiện hành động đi sang trái; ta có thể giải quyết vấn đề này bằng việc phạt chiếc taxi khi rơi vào tình huống đó (tình huống bị "đâm" và tường) và giữ nguyên vị trí hiện tại của nó.

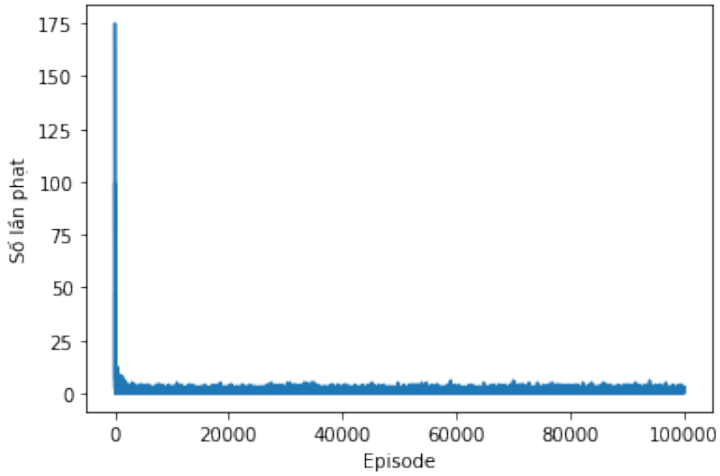
5.1.3 Giải quyết bài toán

Giải pháp được sử dụng để giải quyết bài toán này sẽ là giải thuật Q-Learning. Môi trường của bài toán sẽ được mô phỏng nhờ vào sự trợ giúp của thư viện Gym được cung cấp bởi OpenAI.

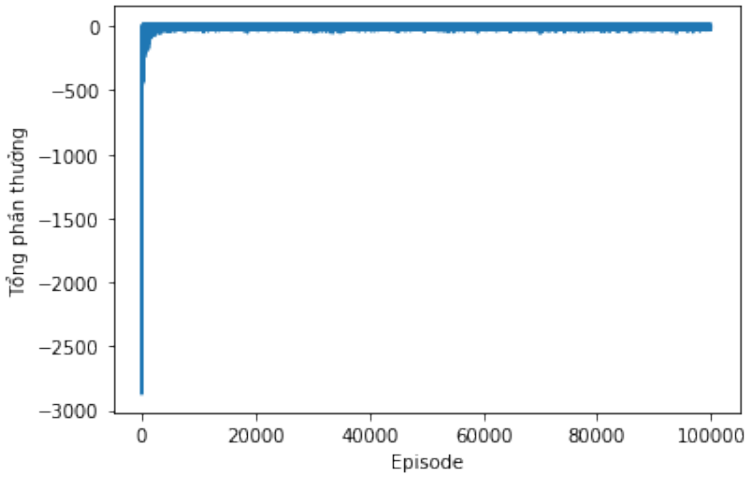
Chiếc taxi sẽ được huấn luyện trong 100000 episode thông qua việc tương tác với môi trường. Một vài kết quả trong quá trình huấn luyện được minh họa như trong Hình 5.2, 5.3 và 5.4.



Hình 5.2: Số bước chuyển chiếc taxi thực hiện tại mỗi episode.

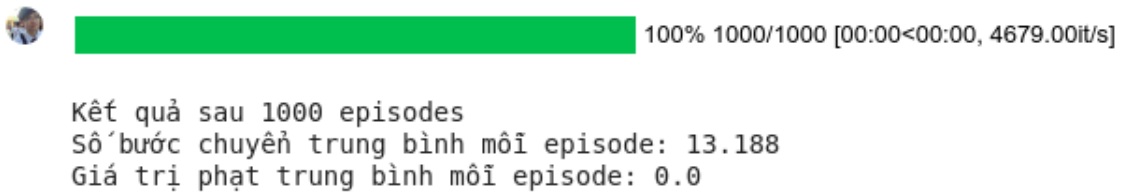


Hình 5.3: Số lần chiếc taxi đón/trả khách sai vị trí tại mỗi episode.



Hình 5.4: Số phần thưởng chiếc taxi nhận được tại mỗi episode.

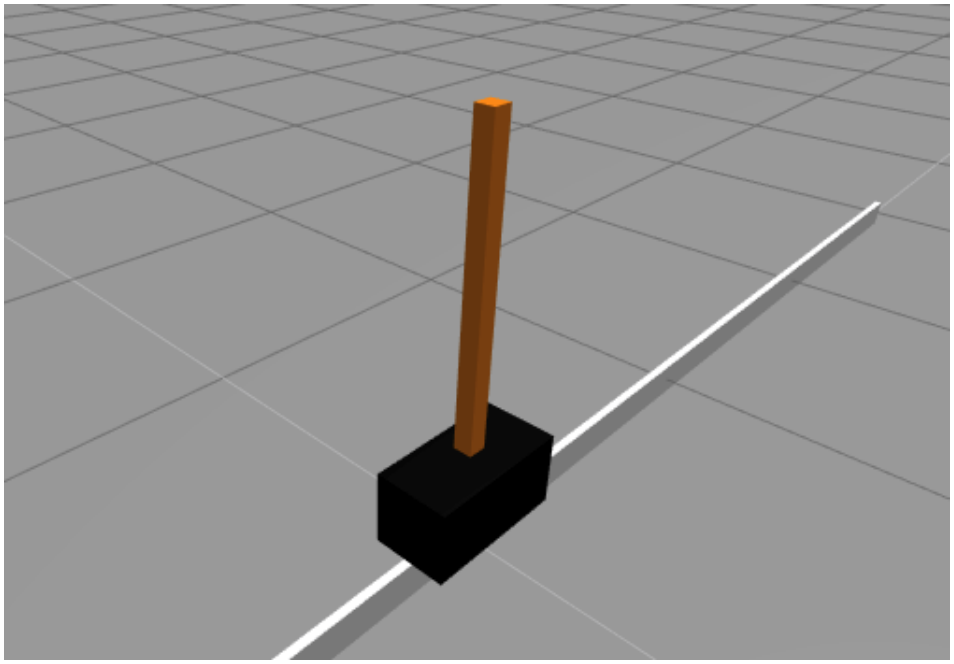
Kết quả chạy của bài toán với 1000 episode sau quá trình huấn luyện được minh họa như trong Hình 5.5



Hình 5.5: Kết quả chạy với 1000 episode.

Có thể thấy chiếc taxi đã được huấn luyện khá tốt, không mắc bất cứ sai lầm nào trong việc đón/trả khách và thực hiện việc chọn đường đi khá "thông minh".

5.2 Bài toán cân bằng con lắc ngược



Hình 5.6: Hình minh họa bài toán cân bằng con lắc ngược.

5.2.1 Bài toán

Ta có một con lắc ngược được gắn phía trên một chiếc xe đẩy bằng một khớp nối có thể quay. Chiếc xe được đặt trên một đường ray không có ma sát và chỉ có thể di chuyển sang trái hoặc sang phải. Ở trạng thái khởi đầu, con lắc được đặt thẳng đứng, hay nói cách khác con lắc và đường thẳng vuông góc với đường ray tạo thành một góc 0°. Mục tiêu của bài toán là không để cho con lắc bị đổ bằng cách tăng hoặc giảm vận tốc của xe đẩy. Trò chơi sẽ kết thúc khi:

- con lắc bị "đổ" (góc của con lắc nhỏ hơn -12°hoặc lớn hơn 12°),
- hoặc vị trí của xe đẩy năm ngoài đoạn $[-2.4, 2.4]$ (tâm của chiếc xe đẩy đi ra ngoài khung nhìn),
- hoặc số bước vượt quá 200.

5.2.2 Mô hình hóa bài toán

Phần thưởng

Hệ chuyển động sẽ nhận được phần thưởng +1 sau mỗi bước (bao gồm cả bước kết thúc).

Không gian trạng thái

Mỗi trạng thái của hệ sẽ được cấu tạo bởi 4 thành phần:

- Vị trí xe đẩy
- Vận tốc xe đẩy
- Góc của con lắc ngược
- Vận tốc ở đỉnh con lắc

Miền giá trị của các thành phần này được minh họa như trong Bảng 5.1.

Chỉ số	Trạng thái	Min	Max
0	Vị trí xe đẩy	-2.4	2.4
1	Vận tốc xe đẩy	-Inf	Inf
2	Góc của con lắc ngược	$\sim$-41.8°	$\sim$41.8°
3	Vận tốc ở đỉnh con lắc	-Inf	Inf

Bảng 5.1: Các trạng thái của hệ.

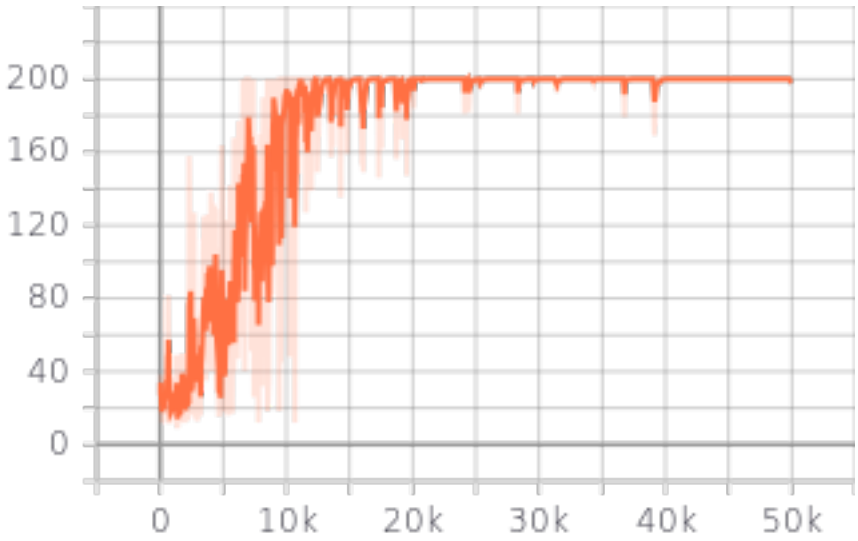
Không gian hành động

Có tổng cộng 2 hành động để di chuyển con lắc, đó là đẩy xe sang trái và sang phải.

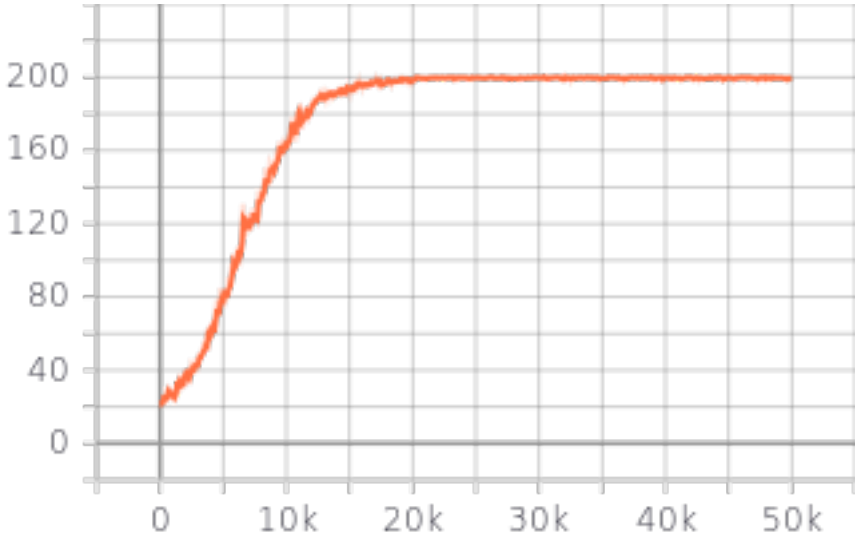
5.2.3 Giải quyết bài toán

Với số trạng thái quá lớn, việc sử dụng giải thuật Q-Learning như trong bài toán "Chiếc taxi thông minh" trước đó trở nên bất khả thi về không gian lưu trữ. Việc sử dụng một mạng nơ-ron nhân tạo (Artificial Neural Network) thay thế cho bảng Q-Table có vẻ phù hợp hơn rất nhiều.

Hệ xe đẩy con lắc sẽ được huấn luyện trong 50000 episode thông qua việc tương tác với môi trường. Một vài kết quả trong quá trình huấn luyện được minh họa như trong Hình 5.7 và 5.8.



Hình 5.7: Số phần thưởng hệ nhận được tại mỗi episode.



Hình 5.8: Số phần thưởng trung bình hệ nhận được qua các episode.

Dựa vào các biểu đồ, ta có thể thấy hệ dễ dàng đạt được số điểm tối đa, 200 điểm, sau khoảng 10000 episode.

Chương 6

Kết luận

Trong báo cáo này, chúng em đã trình bày sơ lược về quá trình Markov, quá trình quyết định Markov và giải thuật Q-Learning. Chúng em đã lựa chọn hai bài toán cơ bản là Chiếc taxi thông minh và Bài toán cân bằng con lắc ngược để minh họa cho việc áp dụng những kiến thức đã được trình bày trong những chương trước cho Học tăng cường.

Các hướng nghiên cứu tiếp theo của đề tài sẽ tập trung vào các biến thể của thuật toán Q-Learning cho những bài toán cụ thể khác trong thực tế.

Tài liệu tham khảo

[1] J. H Andreae, "Learning machines - a unied view," *Encyclopedia of Information Linguistics and Control*, pp. 261 –270, 1969.

[2] R. Bellman, "A markovian decision process," *Indiana Univ. Math. J.*, vol. 6, pp. 679–684, 4 1957, ISSN: 0022-2518.

[3] D. P. Bertsekas, *Dynamic Programming & Optimal Control*. Athena Scientific; 3 edition (1600), 2005.

[4] P. A. Gagniuc, *Markov Chains: From Theory to Implementation and Experimentation*. John Wiley & Sons, 2017, ISBN: 978-1-119-38755-8.

[5] R. A. Howard, *Dynamic programming and Markov processes*. Massachusetts Institute of Technology P.;Wiley, 1960.

[6] M. L. Puterman, *Markov Decision Processes: Discrete Stochastic Dynamic Programming*. John Wiley & Sons, 1994.

[7] R. Sutton and A. Barto, "Reinforcement learning," *Journal of Cognitive Neuroscience*, vol. 11, pp. 126–134, Jan. 1999.

[8] C. Watkins, "Learning from delayed rewards," Jan. 1989.

[9] C. J. C. H. Watkins and P. Dayan, "Q-learning," *Machine Learning*, vol. 8, no. 3, pp. 279–292, 1992, ISSN: 1573-0565. DOI: 10.1007/BF00992698. [Online]. Available: https://doi.org/10.1007/BF00992698.

[10] P. Werbos, "Advanced forecasting methods for global crisis warning and models of intelligence," *General Systems Yearbook*, pp. 25 –38, 1977.

[11] D. White, "Dynamic programming," *San Francisco HoldenDay*, 1969.

[12] I. Witten, "An adaptive optimal controller for discretetime markov environments," *Information and Control*, pp. 286 –295, 1977.

[13] I. Witten and M. Corbin, "Human operators and automatic adaptive controllers: A comparative study on a particular control task," *International Journal of ManMachine Studies*, vol. 5, pp. 75 –104, 1973.