Xây dựng game cờ caro sử dụng Reinforcement Learning

Phạm Trung Hiếu
Mã sinh viên: 20020664
Trường đại học Công Nghệ , Đại học Quốc gia Hà Nội
Hà Nội, 04/11/2023

Tóm tắt: *Bài luận trình bày phương pháp học tăng cường trong game cờ caro. Thuật toán Q-Learning và Deep Q-Network được sử dụng trong việc dạy máy tính cách chơi game. Bên cạnh đó, báo cáo cũng đánh giá việc sử dụng Reinforcement Learning trong bài toán về chơi game đối kháng so với các phương pháp truyền thống khác ...*

Keywords: *Q-Learning, Deep Q-Network, ...*

I. Giới thiệu

Reinforcement Learning (học tăng cường) là một lĩnh vực quan trọng trong trí tuệ nhân tạo. Khác với các phương pháp học truyền thống khác như supervised learning (học giám sát) và unsupervised learning (học không giám sát), Reinforcement Learning sẽ tập trung vào việc học từ tương tác với môi trường để đưa ra các quyết định nhằm tối ưu hóa mục tiêu.

Từ những năm 50, 60 của thế kỷ trước, ý tưởng về Reinforcement Learning đã hình thành bắt nguồn từ các nghiên cứu về supervised learning và unsupervised learning. Năm 1952, Arthur Samuel đề xuất dự án "Men and Machines", nơi ông phát triển một chương trình chơi cờ đánh bằng cách sử dụng hàm đánh giá tương tự (heuristic evaluation function) để ước tính giá trị của các vị trí trên bàn cờ. Đây có thể coi là một trong những dự án đầu tiên sử dụng học tăng cường trong trò chơi. Năm 1989, Chris Watkins giới thiệu thuật toán Q-learning, một phương pháp quan trọng trong Reinforcement Learning. Q-learning dựa trên việc học cách đánh giá các hành động dựa trên các ước tính giá trị của chúng. Vào năm 2013, DeepMind, một công ty nghiên cứu trí tuệ nhân tạo, giới thiệu Deep Q-Network , sự kết hợp của học tăng cường với mạng nơ-ron sâu (Deep Learning). Deep Q-Network gây tiếng vang lớn khi thành công trong việc huấn luyện một hệ thống có thể chơi nhiều trò chơi Atari chỉ bằng cách sử dụng hình ảnh màn hình và điểm số.

Cho đến nay, Reinforcement Learning tiếp tục phát triển nhanh chóng và có ứng dụng rộng rãi trong nhiều lĩnh vực như điều khiển robot, tự động lái xe, quyết định tối ưu và nhiều ứng dụng AI khác. Một trong những điều gây được tiếng vang lớn của Reinforcement Learning là vào năm 2015, AlphaGo, một dự án của DeepMind, đánh bại vô địch cờ vây thế giới Lee Sedol trong một trận đấu 5 ván. AlphaGo sử dụng kỹ thuật học tăng cường kết hợp với deep learning để đạt được điều này. Một phiên bản nâng cấp của AlphaGo là AlphaZero. AlphaZero không chỉ chơi cờ vây mà còn có

khả năng học và chơi nhiều trò chơi khác nhau mà không cần thông tin chuyên môn trước.

Học tăng cường có thể áp dụng được trong các trò chơi cờ và trong bài báo cáo này ta sẽ tiến hành sử dụng hai thuật toán điển hình của Reinforcement Learning là Q-Learning và Deep Q-Network để xây dựng nên một chương trình chơi cờ caro thông minh.

II. Nền tảng lý thuyết

A. Khái niệm cơ bản trong Reinforcement Learning

Agent: Là thực thể hoặc chương trình đang học từ môi trường và ra quyết định để tối ưu hóa mục tiêu.

Environment: là môi trường chứa agent. Nó có thể tương tác được với agent, đáp ứng với hành động của agent và cung cấp thông tin về trạng thái agent và phần thưởng.

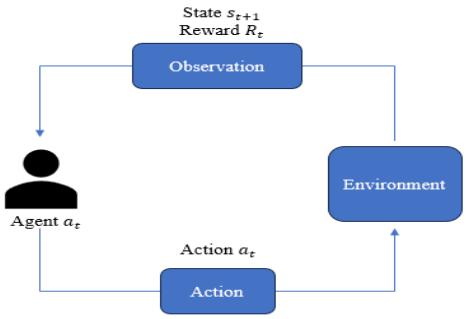
State: Trạng thái biểu diễn tình hình hiện tại của môi trường. Trong nhiều trường hợp, trạng thái cũng bao gồm lịch sử của các trạng thái và hành động trước đó.

Action: Là các lựa chọn có sẵn mà agent có thể thực hiện trong một trạng thái cụ thể. Mục tiêu của agent là chọn hành động tối ưu để tối đa hóa phần thưởng dự kiến.

Reward: Là một số thực hoặc giá trị được cung cấp bởi môi trường sau mỗi hành động của agent. Phần thưởng thường đại diện cho mức độ "hài lòng" của agent với kết quả của hành động.

Policy: Là một quy tắc mà agent sử dụng để chọn hành động trong mỗi trạng thái. Mục tiêu là tìm ra chính sách tối ưu để đạt được mục tiêu tối ưu.

Value Function: Là một hàm đánh giá mức độ "tốt" của một trạng thái hoặc cặp trạng thái-hành động. Các thuật toán học tăng cường cố gắng tối ưu hóa hàm giá trị để đạt được mục tiêu tối ưu.



Hình 1: Các yếu tố trong Reinforcement Learning

B. Thuật toán

1) Markov Decision Process

Markov Decision Process cung cấp một nền tảng toán học cho việc mô hình hóa việc ra quyết định trong các tình huống mà kết quả là một phần ngẫu nhiên và một phần dưới sự điều khiển của một người ra quyết định. Để một bài toán đưa về Markov Decision Process thì các state trong bài toán đó phải thỏa mãn markov property: mỗi state chỉ phụ thuộc vào state trước nó. Từ state s chuyển sang state s' sẽ có xác xuất chuyển đổi được định nghĩa bằng công thức sau: $P_{ss'} = P[S_{t+1} = s' \mid S_t = s]$

2) Q-Learning

QLearning là một thuật toán học tăng cường (reinforcement learning). Nó được sử dụng để học một chiến lược tối ưu cho một tác vụ dựa trên tương tác với môi trường. Thuật toán hoạt động theo các bước sau:

Bước 1: *Khởi tạo bảng ngẫu nhiên bảng Q table*

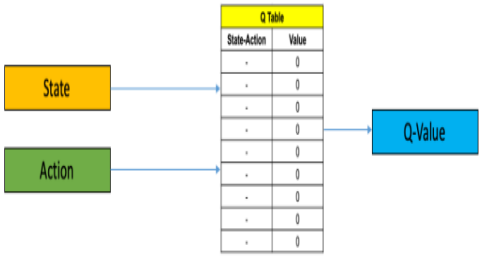
Bước 2: *Khởi tạo môi trường. Tại đây agent có state và có action. Chọn giá trị max trong cặp state-action chuyển qua state mới. Đồng thời môi trường trả lại reward*

Bước 3: *Thực hiện cập nhật bảng Q-table ($Q_{new}[S_t, A_t]$) dựa theo công thức*

$$Q_{new}(s_t, a_t) = (1-\alpha).Q_{oldvalue}(s_t, a_t) + \alpha.(r_t + \gamma.\max_a Q(s_{t+1}, a))$$

Trong đó: α *là learning rate,* γ *là discount factor,* r *là reward*

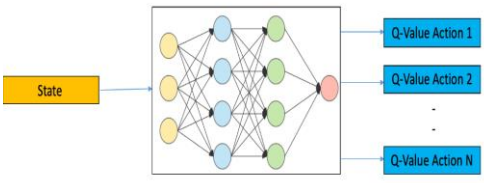
Bước 4: *Thực hiện lặp các bước trên*



Hình 2: Mô tả thuật toán Q-Learning

3) Deep Q-Network

Deep Q-Network được phát triển bởi nhóm DeepMind vào năm 2015. Ý tưởng chính của Deep Q-Network là sử dụng một mạng nơ-ron để xấp xỉ hàm giá trị hành động (Q-function), đại diện cho giá trị dự đoán của mỗi hành động trong một trạng thái cụ thể. Hàm giá trị Q định nghĩa giá trị của mỗi cặp trạng thái-hành động và thể hiện được "chất lượng" của một hành động khi đang ở một trạng thái cụ thể.



Hình 3: Mô tả thuật toán Deep Q-Network

III.Phương pháp giải quyết vấn đề

A. Luật chơi

Luật chơi: Cờ caro (gomoku) là một trò chơi hai người chơi, nơi mục tiêu của mỗi người chơi là tạo ra một dãy liên tiếp gồm 5 quân cờ ("X" hoặc "O") của mình trên một hàng ngang, hàng dọc hoặc đường chéo. Người chơi đầu tiên tạo ra một dãy đủ theo cách trên sẽ chiến thắng. Mỗi người chỉ được đi một nước tại lượt chơi của mình và không được phép đi đè vào vị trí đối thủ đã đi.

B. Mô tả bài toán cờ caro

Để giải quyết bài toán dạy máy tính chơi cờ caro, ta cần xác định cụ thể các yếu tố sau của bài toán:

Agent: là một đối tượng có thể tương tác với environment thông qua quan sát và quyết định hành động của mình thông qua quan sát đó. Đối với game cờ caro, đối thủ chính là player. Player có nhiệm vụ thắng ván đấu. Người chơi thắng khi tạo được dãy liên tiếp 5 ô theo 1 trong các chiều sau chiều dọc, chiều ngang, đường chéo.

State: Sau mỗi action được agent đưa ra thì environment trả về một state. Agent sẽ dựa vào state đó để quan sát và đưa ra hành động tiếp theo. Trong cờ caro, với bàn cờ 20x20 thì state sẽ là 400 ô trong bàn cờ.

Action: Mỗi ô trong bàn cờ sẽ tương ứng với 1 action đánh số từ 0 đến 399. với những ô đã được đánh thì action đó coi như không hợp lệ và được bỏ qua.

Reward: Với mỗi action agent đưa ra thì environment sẽ trả về một reward. Reward có thể được coi là cách mà environment đánh giá agent khi thực hiện một action nào

đó. Mục tiêu của bài toán là tối đa hóa tổng reward thu được. Trong cờ caro, ta sẽ quy định reward sẽ là +1 khi tại đó game kết thúc và agent thắng. Environment sẽ trả về -1 nếu như agent thua. Và các trường hợp còn lại reward trả về là 0. Để agent chọn ra được action thì ta cần có action-value function ký hiệu là $q_\pi(s,a)$. Action-value function cho biết total reward khi chọn action a tại state s.

$$q_\pi(s,a) = E_\pi[G_t \mid S_t = s, A_t = a] = E_\pi[\sum_{k=0}^{\infty} \gamma^k R_{t+k+1} \mid S_t = s, A_t = a]$$

γ: đại diện cho độ quan trọng của reward trong tương lai , giá trị càng nhỏ thì càng ít quan trọng.

C. Thiết lập huấn luyện

1) Q-Learning

Action-value function trong Q-learning là một look-up table gọi là Q-table được khởi tạo ngẫu nhiên. Với mỗi step trong episode thì agent sẽ chọn ra action bằng epsilon-greedy. Epsilon-greedy là thuật toán lựa chọn action dựa trên giá trị epsilon ($0 \leq$ epsilon ≤ 1), đầu tiên thuật toán random một số p($0 \leq p < 1$) nếu $p \leq$ epsilon thì lựa chọn random action và ngược lại chọn greedy action có value lớn nhất. Giá trị epsilon dùng để cân bằng giữa exploration và exploitation nếu agent chỉ chọn greedy action thì có khả năng nó sẽ bỏ qua các action khác có phần thưởng lớn hơn, nếu chỉ toàn chọn random action thì Q-table sẽ hội tụ chậm. Vây trên thực tế thì người ta sẽ khởi tạo epsilon là một còn số lớn như 0.99 chẳng hạn rồi giảm dần giá trị này về một số nhỏ hơn ví dụ như 0.1.

Các action được đưa vào environment, environment trả về state kế tiếp và reward, Q-table được update dựa trên công thức:

$$Q_{new}(s_t, a_t) = (1-\alpha).Q_{oldvalue}(s_t, a_t) + \alpha.(r_t + \gamma.\max_a Q(s_{t+1}, a))$$

2) Deep Q-Network

Deep Q-Network tính action-value ta chỉ việc đưa state vào là input thì neural network lập tức đưa ra ouput là các giá trị action-value. Đối với game cờ caro sẽ tiến hành sẽ sử dụng 2 neural network để tránh tình trạng overfit. Bên cạnh đó, vấn đề của Deep Q-Network gặp phải đó là catastrophic forgetting. Để giảm bớt vấn đề này thì ta sẽ sử dụng 1 lifehack đó là experience replay. Với mỗi step thì ta sẽ có 1 tuple (s, a, r, s', done) được lưu vào memory với done là flag cho ta biết state s' có phải terminal state hay không, sau đó sample 1 mini batch và đưa vào train neural network.

IV. Phân tích và thảo luận

A. So sánh với các thuật toán truyền thống khác

Trong các trò chơi có hai người chơi đối đầu như cờ caro, cờ vua, cờ tướng, … đã tồn tại một vài phương pháp giải. Min-Max là một trong số đó. Minmax sẽ coi đối thủ là một người chơi hoàn hảo, công việc của người chơi khi này là tối ưu hóa điểm thưởng của bản thân và tối thiểu hóa điểm thưởng của đối thủ. Tuy nhiên điểm hạn chế của phương pháp này nằm ở chỗ, không phải đối thủ nào cũng là hoàn hảo, nó giả định rằng cả hai người chơi đều đi nước đi tối ưu và không xem xét các hành động ngẫu nhiên hoặc sai lầm của đối thủ. Đồng thời thuật toán chỉ tốt trong điều kiện không gian trạng thái nhỏ.

Nhược điểm trên của Min-Max đã được Reinforcement Learning khắc phục. Trong

Reinforcement Learning, ta không cần phải giả định người chơi là một đối thủ hoàn hảo. Thay vào đó, mô hình sẽ tự học thông qua những kinh nhiệm thu được từ quá trình tương tác với môi trường. Hai thuật toán chính được thảo luận trong báo cáo này là hai thuật toán điển hình của Reinforcement Learning. Q-Learning sẽ phù hợp với bài toán có số lượng state nhỏ. Trong khi đó Deep Q-Network sẽ phù hợp với bài toán có không gian trạng thái lớn. Trong các trò chơi về cờ, thường đòi hỏi không gian tìm kiếm lớn cũng như tồn tại nhiều trạng thái. Do vậy Deep Q-Network sẽ là sự lựa chọn phù hợp.

B. Đề xuất phương pháp cải thiện Deep Q-Network

1) Double Deep Q-Network

Ý tưởng cơ bản của Double Deep Q-Network là sử dụng hai mạng nơ-ron (neural networks) riêng biệt: một mạng chính được sử dụng để dự đoán giá trị hành động (Q-values), và một mạng phụ (target network) được sử dụng để tạo ra các giá trị target mục tiêu cho việc cập nhật mô hình.

Mỗi một khoảng thời gian (ví dụ: sau mỗi bước huấn luyện), các trọng số của mạng chính được sao chép vào mạng phụ. Điều này giúp ổn định quá trình học, vì mạng phụ không bị thay đổi trong quá trình cập nhật, và nó tạo ra một ước lượng đáng tin cậy hơn cho giá trị target. Bằng cách sử dụng Double Deep Q-Network, người ta có thể cải thiện sự ổn định và hiệu suất của mô hình so với Deep Q-Network cơ bản, đặc biệt trong các tình huống mà overestimation bias có thể gây ra vấn đề.

2) Prioritized Experience Replay (PER)

Trong quá trình huấn luyện, các mô hình học tăng cường sử dụng một bộ nhớ đệm (replay buffer) để lưu trữ các trạng thái (states), hành động (actions), phần thưởng (rewards), và các trạng thái kế tiếp (next states) từ các tương tác trước đó với môi trường. Điều này giúp mô hình học từ những trải nghiệm trước đó và làm cho việc học trở nên ổn định hơn.

PER điều chỉnh quá trình lấy mẫu từ replay buffer bằng cách gán một ưu tiên cho các trải nghiệm (samples) dựa trên mức độ quan trọng của chúng đối với quá trình học. Trong PER, mỗi trải nghiệm được gán một mức độ ưu tiên (priority) được tính toán dựa trên sự sai lệch giữa ước lượng Q-value và giá trị target cho mỗi hành động.

Các mẫu có mức độ ưu tiên cao sẽ được lấy mẫu nhiều hơn, tập trung vào việc học từ các trải nghiệm quan trọng. Điều này giúp mô hình học nhanh hơn và cải thiện hiệu suất học tập.

Khi sử dụng PER, cần cẩn trọng với việc cập nhật mức độ ưu tiên để tránh sự không ổn định trong quá trình học. Có một số kỹ thuật như áp dụng một mức độ bias (nghiêng) vào các mẫu được lấy mẫu dựa trên mức độ ưu tiên để giảm thiểu các vấn đề liên quan đến việc cập nhật mức độ ưu tiên.

3) Dueling Deep Q-Network

Ý tưởng cơ bản của Dueling Deep Q-Network là tách biệt việc ước lượng giá trị hành động (action values) thành hai phần: một phần dự đoán giá trị của trạng thái (state value) và một phần dự đoán sự ưu tiên của từng hành động trong trạng thái đó

(advantage values). Bằng cách làm điều này, Dueling Deep Q-Network giúp mô hình tập trung vào các hành động quan trọng trong mỗi trạng thái.

V. Kết luận

Trong báo cáo này đã trình bày hai thuật toán Q-Learning và Deep Q-Network sử dụng cho game cờ caro. Hai thuật toán này có những điểm mạnh và điểm hạn chế khác nhau. Đối với Q-Learning, đây là thuật toán dễ triển khai hiệu quả trong các môi trường với không gian trạng thái hữu hạn và không thể áp dụng trong môi trường liên tục. Deep Q-Network có khả năng xử lí trong miền không gian lớn, liên tục thông qua mạng noron. Tuy nhiên, việc chọn Q-Learning hay Deep Q-Network sẽ phụ thuộc vào điều kiện của từng bài toán cụ thể.

VI. Tài liệu tham khảo

1. CodeLearn (no date) Dạy Máy Chơi tictactoe bằng deep learning, CodeLearn. Available at: https://codelearn.io/sharing/day-ai-danh-tictactoe-voi-deep-learning?fbclid=IwAR29xrlKBE7wJD9G9y6SAM_gB_TLEwzbH0Jn8q4hS8pOXGbwyz8NPBCDyh8 (Accessed: 04 November 2023).

2. Tùng, alexblack2202@gmail. comPhạm D. (2020) Reinforcement learning VÀ Tictactoe, Phạm Duy Tùng Machine Learning Blog. Available at: https://www.phamduytung.com/blog/2020-12-26---tic-tac-toe/?fbclid=IwAR3843Frq9ADBhS29OS3qkmaomhzRA4fYvT-dP3Gg7PrZ4V1Z76-pKRNbsI (Accessed: 04 November 2023).

3. Deep Q-Network (DQN) – Học Tăng Cường (no date) Bài viết. Available at: https://tek4.vn/deep-q-network-dqn-hoc-tang-cuong-phan-5 (Accessed: 05 November 2023).

4. Michael Kearns and Satinder P. Singh. Finite-sample convergence rates for Q-learning and indirect algorithms. In M.J. Kearns, S.A. Solla, and D.A. Cohn, editors, Advances in Neural Information Processing Systems 11, pages 996–1002, 1999.

5. C. Szepesvari. The asymptotic convergence-rate of Q-learning. In M.I. Jordan, M.J. Kearns, and S.A. Solla, editors, Advances in Neural Information Processing Systems 10, pages 1064–1070, 1998.