Tạp chí Khoa học Đại học Huế: Kỹ thuật và Công nghệ; ISSN 2588–1175

Tập 127, Số 2A, 2018, Tr. 147–157; DOI: 10.26459/hueuni-jtt.v127i2A.5099

KHẢO SÁT VÀ ĐÁNH GIÁ VỀ CÁC HƯỚNG TIẾP CẬN LỰA CHỌN ĐẶC TRƯNG TRONG BÀI TOÁN ĐÁNH CỜ CÓ ĐỘ PHÂN NHÁNH CAO

Đặng Công Quốc[1], Nguyễn Đăng Bình[1], Nguyễn Quốc Huy [2]

[1] Trường Đại học Khoa học, Đại học Huế
77 Nguyễn Huệ, phường Phú Nhuận, TP. Huế, tỉnh Thừa Thiên Huế
[2] Khoa Công nghệ thông tin – Trường Đại học Sài Gòn
273 An Dương Vương, Quận 5, TP. Hồ Chí Minh

Tóm tắt. Lựa chọn đặc trưng đóng vai trò quan trọng trong học máy. Các chương trình đánh cờ là môi trường thử nghiệm tuyệt vời cho các nghiên cứu về AI, đây thực sự là thách thức lớn khi trò chơi có độ phân nhánh cao như cờ Vây, Amazon, Connect6. Tìm đặc trưng tốt từ dữ liệu các ván cờ có sẵn thật sự là vấn đề không dễ dàng. Bài báo này trình bày những vấn đề cốt lõi và quan trọng trong việc phát triển một chương trình đánh cờ như Cây tìm kiếm, hàm lượng giá, lựa chọn đặc trưng, và làm cách nào để đánh trọng số cho các đặc trưng dựa vào dữ liệu các ván cờ có sẵn.

Từ khóa: lựa chọn đặc trưng, Connect6, hàm lượng giá

1 Giới thiệu

Đánh cờ là một chuỗi lặp đi lặp lại việc chọn lựa nước đi giữa hai người chơi. Trạng thái bàn cờ thay đổi khi một nước đi mới được thực hiện. Nói cách khác, đây là bài toán tìm kiếm giải pháp tối ưu trên một trạng thái của bàn cờ. Mức độ tối ưu của việc chọn lựa giải pháp thể hiện tính thông minh của chương trình.

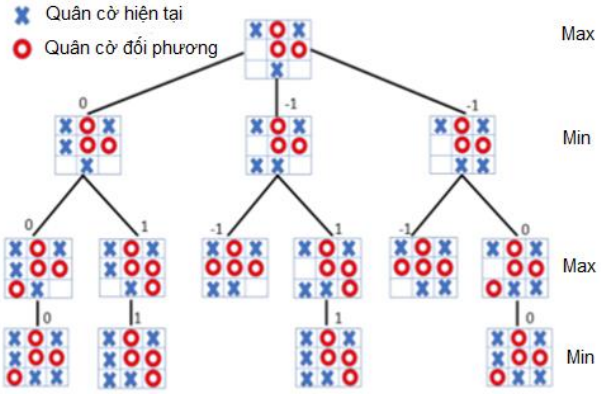
Một cây trò chơi bao gồm tất cả các nước đi có thể có của hai người chơi và mỗi nút của cây thể hiện một trạng thái bàn cờ sau khi nhận một nước đi từ người chơi. Từ một nút (trạng thái) hiện tại có thể có nhiều lựa chọn cho nước đi tiếp theo đó; số nước có thể chọn được gọi là hệ số phân nhánh. Độ sâu của cây trò chơi là số lần thay đổi lượt đi của hai người chơi. Hình 1 minh họa cây trò chơi của trò chơi đối kháng Tic-Tac-Toe; trò chơi này cực kì đơn giản vì chơi trên không gian 3 × 3 = 9 ô. Hai người chơi là X và O. Đối với trò chơi Tic-Tac-Toe, mỗi ô có tối đa 3 trạng thái (O, X, trống). Số ô của bàn cờ là 9, nên không gian trạng thái bàn cờ của trò chơi Tic-Tac-Toe là 3^9 = 19.683. Số lượng cây là 9! = 362.880.

Đối với máy tính hiện đại thì những trò chơi có không gian tìm kiếm nhỏ như trò chơi Tic-Tac-Toe thì máy tính có thể vét cạn, và lúc đó chương trình đánh cờ chỉ từ hòa đến thắng vì biết được nước đi tốt nhất theo cách đi của đối phương. Các trò chơi có không gian tìm kiếm trung bình như Connect4, Riversi, Chess, Chinese Chess và Shogi thì máy tính không đủ khả

** Liên hệ: dangcongquoc1968@gmail.com*

Nhận bài: 4–9–2018; Hoàn thành phản biện: 18–10–2018; Ngày nhận đăng: 30–01–2019

năng để vét cạn. Lúc đó, máy tính có thể tính trước một số bước nào đó rồi ước lượng. Chương trình máy tính mạnh hay yếu nhờ vào khả năng ước lượng.



Hình 1. Cây trò chơi Tic-Tac-Toe

Một số thuật toán tìm kiếm trên cây truyền thống như tìm kiếm Minimax, Alpha-Beta ($\alpha\beta$) và tìm kiếm A* đã được ứng dụng rất thành công trong nhiều lĩnh vực. Ví dụ trong trò chơi, bắt đầu từ trạng thái hiện hành của một trò chơi, cây tìm kiếm được vẽ ra để miêu tả các nước đi có thể từ trạng thái hiện hành đó. Nếu ta mở rộng cây cho đến nước đi cuối cùng thì chắc chắn sẽ tìm ra được nước đi tối ưu theo kiểu lan truyền ngược Minimax.

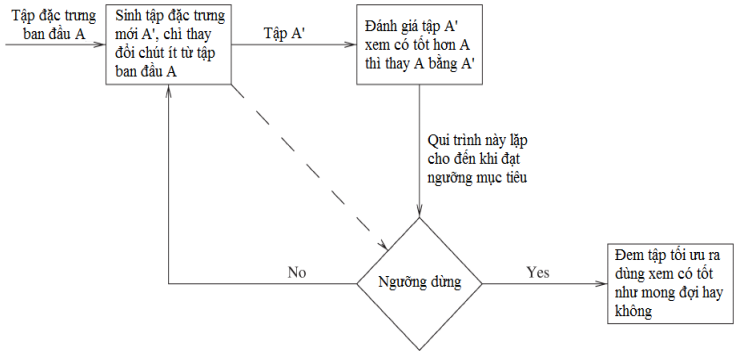
Connect6 là một trò chơi có tính chất đối kháng và được chơi trên một bàn cờ có kích thước 19 × 19 là họ trò chơi k-in-a-row [1] do Xu và cs. đề xuất vào năm 2013. Kích thước bàn cờ lớn và luật chơi với hai quân cờ mỗi lượt nên không gian tìm kiếm nước đi của Connect6 rất lớn, độ phức tạp của không gian trạng thái là 10.172 nên phải lựa chọn đặc trưng để tìm nước đi tối ưu, từ đó tạo tiền đề để tìm ứng viên tiềm năng dẫn đến chiến thắng.

2 Các phương pháp lựa chọn đặc trưng

Trong phương pháp học máy, thay vì phải học hết tập dữ liệu huấn luyện lớn với chi phí cao và không hiệu quả do dữ liệu có những yếu tố dư thừa và nhiễu. Để kết quả huấn luyện cao thì thông thường học qua các đặc trưng thay vì học nguyên tập dữ liệu huấn luyện [3]. Số lượng đặc trưng (features) càng nhiều thì độ chính xác càng cao; ngược lại, lượng đặc trưng quá nhiều sẽ khiến cho quá trình huấn luyện và quá trình phân loại mất nhiều thời gian hơn. Ngoài ra, nó còn khiến chương trình chiếm dung lượng bộ nhớ và đĩa cứng nhiều hơn. Vì vậy, phải có phương pháp lựa chọn đặc trưng tối ưu, không nhất thiết phải chọn hết tất cả đặc trưng.

Bài toán đặt ra trong phương pháp học máy là phải lựa chọn từ tập các đặc trưng ra một tập con nhỏ hơn mà vẫn đảm bảo độ chính xác của quá trình phân loại. Việc lựa chọn đó được

gọi là lựa chọn đặc trưng. Đối với từng phương pháp học máy, sẽ có những phương pháp tương ứng hiệu quả riêng với nó. Nói cách khác, không có phương pháp nào là tốt nhất. Phương pháp tìm tập đặc trưng phổ biến nhất được mô tả như trong Hình 2.



Hình 2. Quy trình lựa chọn đặc trưng [3]

Có ba hướng tiếp cận tổng quát đối với lựa chọn đặc trưng. Thứ nhất, *hướng tiếp cận lọc* khai thác các thuộc tính chung của dữ liệu huấn luyện độc lập với thuật toán khai phá. Hướng này thông thường đề xuất một độ đo và đo từng đặc trưng riêng biệt và những đặc trưng nào thỏa mãn độ đo thì được chọn. Tuy nhiên, những đặc trưng được cho là tốt theo hướng lọc đôi khi không tốt khi kết hợp. Nói cách khác, nhiều đặc trưng tốt chưa chắc bổ sung cho nhau để cho ra một hàm đánh giá tốt. Thứ hai, *hướng tiếp cận đóng gói* khám phá mối quan hệ giữa lựa chọn tập con đặc trưng thích hợp và tối ưu. Nó tìm kiếm tập con đặc trưng tối ưu đưa vào thuật toán khai phá cụ thể. Những đặc trưng này nếu đo theo công thức đánh giá của hướng filter (theo từng đặc trưng riêng biệt) nhiều khi không đạt ngưỡng và không được chọn. Tuy nhiên, nếu tập đặc trưng này nằm trong một hàm đánh giá thì từng đặc trưng này lại bổ sung cho nhau hiệu quả. Thứ ba, *hướng tiếp cận nhúng* là phương pháp hồi qui cho mô hình tuyến tính được tổng quát hóa. Hướng này thường thêm những giá trị cộng thêm cho hàm đánh giá nhằm giảm tính quá khớp của mô hình (tăng chất lượng của mô hình). Một số thuật toán như LASSO và cây quyết định thuộc phương pháp này.

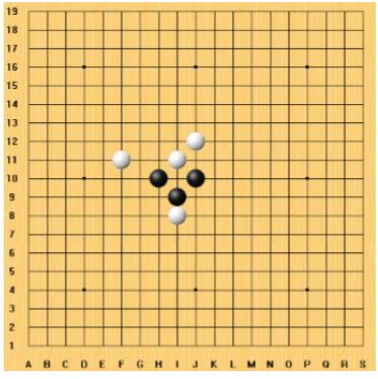
Theo qui trình lựa chọn đặc trưng như mô tả thì các phương pháp tối ưu ngẫu nhiên như Leo đồi, Luyện thép và Di truyền thường được dùng để thiết kế mô hình chọn lựa đặc trưng [4]. Công việc lớn nhất trong phần này là xây dựng một hàm mục tiêu phù hợp cho các phương pháp tối ưu ngẫu nhiên và phương pháp đánh giá kết quả.

3 Cờ Connect6

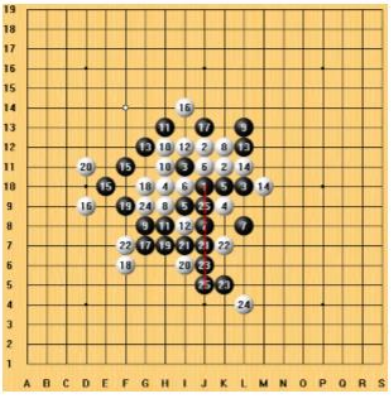
Connect(m, n, k, p, q) ký hiệu họ trò chơi k-in-a-row. Có hai người chơi: trắng và đen. Người chơi thứ nhất với quân đá đen đặt q hòn đá cho di chuyển lần thứ nhất. Sau đó người chơi thứ hai đặt q hòn đá trên bàn $m \times n$ trong mỗi lần. Người chơi nhận được k hòn đá liên tiếp

đầu tiên thì thắng. Connect(m, n, 6, 2, 1) gọi là Connect6 [1,2]. Đầu tiên, người chơi đặt duy nhất một quân đen trên bàn 19 × 19, và sau đó hai người chơi luân phiên đặt hai quân cờ vào bàn này. Bàn cờ Connect6 như Hình 3 đánh số thứ tự theo các dòng từ dưới lên trên bắt đầu từ số 1 đến số 19 và các cột được đánh theo bảng chữ cái alphabet từ trái qua phải bắt đầu từ chữ A đến chữ S. Vị trí giao nhau giữa dòng và cột và chưa có quân cờ nào đặt lên thì vị trí này được gọi là vị trí đặt quân cờ hợp lệ (vị trí này còn được gọi là điểm giao hợp lệ). Vì kích thước bàn cờ là 19 × 19 nên số lượng điểm giao tương ứng để đặt quân là 316 và mọi giao điểm có ba trạng thái (trống, trắng và đen) nên độ phức tạp trạng thái của cờ Connect6 xấp xỉ 3^{361}.

Ván cờ kết thúc khi một trong hai người chơi giành được chiến thắng hoặc các quân cờ đã lấp đầy bàn cờ (không thể đặt quân cờ hợp lệ lên bàn cờ). Người giành chiến thắng là người có được một hàng (chéo, ngang, dọc) gồm 6 quân liên tiếp của mình trước người chơi thứ hai và người chiến thắng được xem là đã thực hiện được một Connect6 [1]. Như trong Hình 4 người chơi cầm quân Đen đã kết thúc ván cờ bằng một chiến thắng trước đối thủ là người chơi quân Trắng.



Hình 3. Bàn cờ trò chơi Connect6



Hình 4. Ván cờ kết thúc dưới một chiến thắng của người chơi quân Đen

4 Các nghiên cứu gần đây về lựa chọn đặc trưng trong bài toán đánh cờ có độ phân nhánh cao

Trong bài báo [6], các tác giả mới chỉ sử dụng hai phương pháp tối ưu ngẫu nhiên: giải thuật leo đồi (Hill-Climbing) và giải thuật luyện thép (Simulated annealing) để tối ưu hóa các đặc trưng của bàn cờ Othello. Họ kết hợp với phương pháp học có giám sát Bradley-Terry Minorization-Maximization (bao gồm mô hình Bradley-Terry và giải thuật Minorization-Maximization) để tìm ra những đặc trưng tốt để sử dụng trong cây tìm kiếm Monte Carlo (MCTS: Monte Carlo Tree Search). Với phương pháp này, nhà nghiên cứu có thể xây dựng được hàm lượng giá hành động (action valuation function) tốt để đánh giá các nước đi hứa hẹn giúp cho máy tính có thể chọn lựa được nước đi tốt nhất có thể trong một thời gian nhất định. Bên cạnh đó, các tác giả còn đưa ra phương pháp thống kê để tìm ra các đặc trưng và đánh giá độ tin cậy các đặc trưng đó trước khi học. Kết quả của các phương pháp này đã áp dụng rất tốt cho cờ Othello.

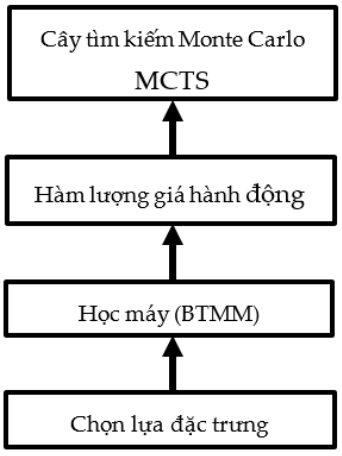
Công trình của Huang [7] đã đưa ra một số phương pháp Heuristic mới cho MCTS tập trung vào hai đóng góp: Thứ nhất, áp dụng thành công giải thuật cân bằng giả lặp ngẫu nhiên (Simulation Balancing – SB) để huấn luyện các tham số cho việc giả lặp ngẫu nhiên trên bàn cờ Vây kích thước 9 × 9. Giải thuật SB do Silver và Tesauro [8] đưa ra năm 2009. Đây là giải thuật học tăng cường nhưng chỉ áp dụng cho bàn cờ có kích thước nhỏ. Một số thí nghiệm đã tiến hành để chứng minh tính hiệu quả trên bàn cờ Vây kích thước 9 × 9 và đã chỉ ra giải thuật SB vượt qua giải thuật học có giám sát nổi tiếng Minorization-Maximization (MM) khoảng 90 Elo. Một số thí nghiệm khác được tiến hành cho cờ Vây kích thước 19 × 19. Kết quả chỉ ra rằng các giải thuật quản lý thời gian thông minh có thể được xem xét để cải thiện sức mạnh khi chơi trò chơi.

Công trình của Loos [9] đã khám phá khả năng kết hợp của nhiều kỹ thuật học máy để thử nghiệm trí tuệ nhân tạo cho các trò chơi loại k-in-a-row. Các kỹ thuật sử dụng gồm Cây quyết định (Decision Trees), Random Forest (bao gồm cây quyết định), giải thuật Minimax và giải thuật di truyền. Trong đó, giải thuật di truyền đóng vai trò chủ đạo để xây dựng trí tuệ cho máy tính. Trong bước đánh giá, giải thuật Minimax tìm kiếm trên cây được sử dụng, mỗi nước dự kiến sẽ có một Random Forest gắn vào được sử dụng như hàm heuristic trong Minimax. Mục đích chính là huấn luyện để tiến hóa các Random Forest tốt nhất có thể. Thí nghiệm trên trò chơi Tic-Tac-Toe, Connect4 và Gomoku trên bàn cờ kích thước 10 × 10 cho kết quả tốt. Tuy nhiên, kết quả thí nghiệm cho thấy phương pháp đưa ra chạy chậm trên kích thước bàn cờ lớn; để cải thiện được tốc độ đòi hỏi tốc độ xử lý cao của CPU. Wu và Chang sử dụng hàm lượng giá trạng thái trên cây tìm kiếm Alpha-Beta dựa vào đặc điểm về mối đe dọa trên cờ Connect6 để xây dựng các vùng phù hợp các quân cờ cần phải đặt để có được trạng thái tốt nhất. Các thành phần trong bài toán đánh cờ bao gồm: Cây tìm kiếm Alpha-Beta, Hàm lượng giá trạng thái, Vùng đặc trưng phù hợp để xây dựng hàm lượng giá.

Yen và Yang [10, 11] sử dụng một phương pháp giả lập mới trong cây tìm kiếm Monte Carlo. Ý tưởng chính đề xuất một biến thể mới của MCTS là sử dụng cây tìm kiếm And/Or kết hợp với phương pháp giả lập lấy mẫu ngẫu nhiên của Monte Carlo. Yen đã sử dụng một khái niệm là vùng phù hợp được kế thừa và phát triển từ Wu cho chương trình đánh cờ của ông. Yen và các cộng sự đã xử lý cho vùng phù hợp chi tiết hơn so với vùng phù hợp của Wu đã đưa ra trước đó và vận dụng vào giai đoạn 2 của MCTS. Hàm lượng giá trong chương trình này là hàm lượng giá hành động và được hỗ trợ bởi vùng phù hợp được xác định qua giải pháp T2, giải pháp TSS để giới hạn không gian trên cây And/Or. Các thành phần trong bài toán đánh cờ bao gồm Cây tìm kiếm And/Or, hàm lượng giá hành động và Vùng đặc trưng để xây dựng hàm lượng giá (đơn nguy cơ và đôi nguy cơ).

5 Mô hình đề xuất phù hợp với cây tìm kiếm Monte Carlo

Trong một số trò chơi có độ phân nhánh cao, những cây tìm kiếm như Minimax và Alpha-Beta thường không phù hợp và xử lý rất chậm vì không gian tìm kiếm quá lớn. Cây tìm kiếm Monte Carlo là phương pháp lấy mẫu dựa trên phương pháp cân bằng giữa việc khai thác và khám phá để tập mẫu tuy nhỏ nhưng đại diện chính xác được không gian tìm kiếm lớn (tránh được việc vét cạn). Cây tìm kiếm Monte Carlo theo lý thuyết thì không cần hàm lượng giá [5]. Tuy nhiên, xây dựng được hàm lượng giá phù hợp sẽ giúp cho việc hội tụ tập mẫu sẽ nhanh hơn. Hàm lượng giá phù hợp cho cây tìm kiếm Monte Carlo thường là hàm lượng giá hành động (khác với hàm lượng giá trạng thái, thường phù hợp với cây tìm kiếm Minimax và Alpha-Beta). Khi đã cần xây dựng hàm lượng giá thì việc lựa chọn đặc trưng là cần thiết. Những đặc trưng được đánh giá là phù hợp hay không thì phải có phương pháp đánh trọng số. Một số nghiên cứu cho thấy phương pháp huấn luyện trọng số dựa trên những ván cờ có sẵn bằng phương pháp BTMM (Bradley-Terry Minorization Maximization) là rất hiệu quả. Hình 5 mô tả mối quan hệ giữa BTMM và cây tìm kiếm Monte Carlo.



Hình 5. Các thành phần chính trong bài toán đánh cờ có độ phân nhánh cao

```
1   public class MonteCarloTreeSearch {
2       static final int WIN_SCORE = 10;
3       int level;
4       int opponent;
5       public Board findNextMove(Board board, int playerNo) {
6           opponent = 3 - playerNo;
7           Tree tree = new Tree();
8           Node rootNode = tree.getRoot();
9           rootNode.getState().setBoard(board);
10          rootNode.getState().setPlayerNo(opponent);
11          while (System.currentTimeMillis() < end) {
12              Node promisingNode = selectPromisingNode(rootNode);
13              if (promisingNode.getState().getBoard().checkStatus()
14                == Board.IN_PROGRESS) {
15                  expandNode(promisingNode);
16              }
17              Node nodeToExplore = promisingNode;
18              if (promisingNode.getChildArray().size() > 0) {
19                  nodeToExplore = promisingNode.getRandomChildNode();
20              }
21              int playoutResult = simulateRandomPlayout(nodeToExplore);
22              backPropogation(nodeToExplore, playoutResult);
23          }
24          Node winnerNode = rootNode.getChildWithMaxScore();
25          tree.setRoot(winnerNode);
26          return winnerNode.getState().getBoard();
27      }
28  }
```

Hình 6. Thuật toán Monte Carlo Tree Search

Trong cây tìm kiếm Monte Carlo như Hình 6 có 4 giai đoạn: Chọn lựa một nút hứa hẹn nhất trong cây theo phương pháp cân bằng giữa Khai thác và Khám phá (hàm selectPromisingNode() trong dòng mã số 12. Giai đoạn tiếp theo là tăng trưởng cây bằng cách Mở rộng một nút con trong nút hứa hẹn được chọn ở giai đoạn trên, và bước mở rộng này được thực hiện ngẫu nhiên (như phương thức getRandomChildNode() trong đoạn mã dòng 19). Giai đoạn tiếp theo là Giả lặp ván cờ từ nút mới được mở rộng và có kết quả thắng thua (hàm simulateRandomPlayout() trong đoạn mã dòng 21). Giai đoạn cuối là lan truyền ngược kết quả thắng thua đó lên nút hứa hẹn ở giai đoạn 1 (hàm backpropagation () trong dòng mã 22). Quá trình này được gọi là 1 playout. Trong thời gian cho phép (vòng lặp trong đoạn mã số 11) việc thực hiện playout cứ thực hiện. Số lần playout được thực hiện coi như là mẫu được lấy tại nút hứa hẹn.

Trong việc lấy ngẫu nhiên, vai trò của các đặc trưng γ_i được thực hiện thông qua công thức (2).

$$p(m_j) = \frac{\prod_{feature\ i \in m_j} \gamma_i}{\sum_{legal\ moves m} \left(\prod_{feature\ i \in m_j} \gamma_i \right)} \quad (1)$$

$$\gamma_i \leftarrow \frac{W_i}{\sum_{j=1}^{N} \frac{C_{ij}}{E_j}} \quad (2)$$

$$MLE = \frac{\sum_{i \in m} \left(log(prob(m_i)) \right)}{N} \quad (3)$$

Công thức (1) được áp dụng trong giai đoạn Mở rộng và Giả lặp. Thay vì chọn ngẫu nhiên thuần túy, chúng ta có thể chọn theo phương pháp Roullet Wheel (bánh xe may mắn) và nước đi nào có đặc trưng tốt thì có tỷ lệ chọn lựa cao hơn. Ngoài ra, trong giai đoạn Chọn lựa, thay vì chọn nút hứa hẹn theo công thức (4) thuần túy, chúng ta có thể chọn theo công thức (5) có lệch theo yếu tố đặc trưng. Trong công thức (4) và (5), n là tổng số playouts trong khoảng thời gian suy nghĩ trong đoạn mã 11 (Hình 6). Trong đó, n_j là số lần playouts qua nút thứ j, và w_i là số lần đặc trưng i xuất hiện trong nút j. Trong công thức (5), K là hệ số lệch; trong một số thí nghiệm thì K lớn gấp 5 lần n.

$$UCT_j = \frac{w_i}{n_j} + C\sqrt{\frac{\ln n}{n_j}} \quad (4)$$

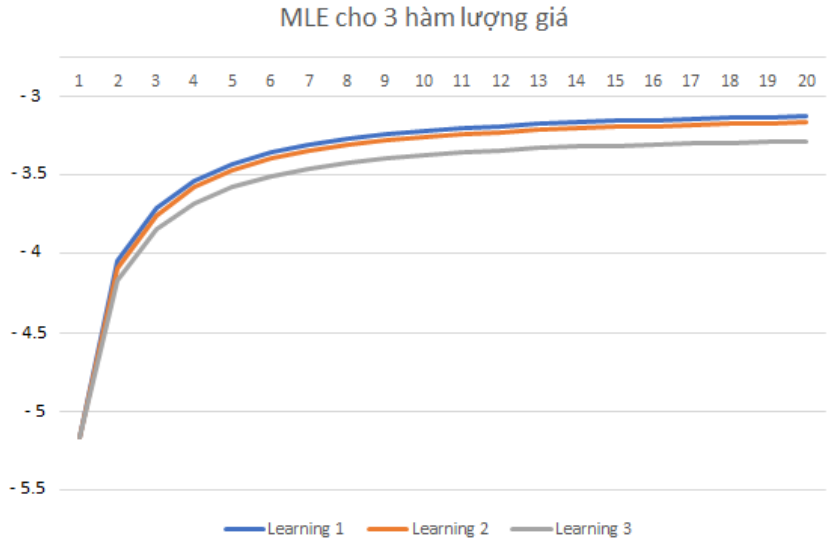
$$UCTbias_j = \frac{w_i}{n_j} + C\sqrt{\frac{\ln n}{n_j}} + C_{BT}\sqrt{\frac{K}{n+K}}P(m_j) \quad (5)$$

Phương pháp học máy BTMM áp dụng mô hình Bradley-Terry vào phương pháp tối ưu Minorization Maximization. Công thức tối ưu (2) do Remi Coulom đề xuất năm 2007 [6] cũng áp dụng suy diễn Bayes để tối ưu đặc trưng từ tập dữ liệu có sẵn. Sau khi xác định được trọng số của từng đặc trưng thì việc xây dựng hàm lượng giá dựa trên đặc trưng là việc dễ dàng. Hàm lượng giá hành động thông thường là tích các trọng số đặc trưng liên quan đến hàm lượng giá. Một hàm lượng giá tốt sẽ giúp cho cây tìm kiếm Monte Carlo rút ngắn thời gian hội tụ trong việc tìm khả năng tốt nhất (nước đi tối ưu) trên một trạng thái bàn cờ hiện hành.

6 Thí nghiệm ban đầu

Thí nghiệm trên 1.000.000 ván cờ Connect6 có chất lượng cao và thử nghiệm trên các tập 4 mẫu có độ dài 6, 4 mẫu có độ dài 7, 4 mẫu có độ dài 8, và 4 mẫu trong đó 2 mẫu có độ dài 8 và 2 mẫu có độ dài 7. Mỗi đặc trưng được đánh trọng số bằng công thức (4) và dùng độ đo MLE để xác định loại mẫu nào phù hợp nhất cho các vị trí trên bàn cờ. Trong 1.000.000 ván cờ, 995.000 ván được làm dữ liệu huấn luyện, 5.000 ván được làm dữ liệu kiểm tra theo phương pháp huấn luyện BTMM với số vòng lặp 20 cho tất cả các thí nghiệm. Việc thí nghiệm có 2 giai đoạn: Giai đoạn 1 dùng độ đo MLE như công thức (3) để xác định mẫu phù hợp để xây dựng

hàm lượng giá. Giai đoạn 2 xây dựng hàm lượng giá cho chương trình VN-Connect để đấu với chương trình X6, một chương trình rất mạnh từng đoạt giải nhất quốc tế ICGA Computer Olympiad.



Hình 7. So sánh giá trị learning giữa 3 hàm lượng giá

Hàm lượng giá 4 mẫu có độ dài 8 có hiện tượng quá khớp (overfitting) nên bị loại trừ. Những hàm lượng giá còn lại là tốt nhất (ứng với giá trị Learning 1 như trong Hình 7). Trong hình này, trục tung là giá trị MLE (Mean Log-Evidence) cũng được áp dụng trong phương pháp kiểm tra chéo với tập dữ liệu dùng để huấn luyện và đánh giá là tập các ván cờ Connect-6 được thu thập. Trục hoành mô tả số lần lặp để tối ưu giá trị trọng số của đặc trưng như công thức (2).

Dựa trên thí nghiệm giai đoạn 1, nhóm tác giả xây dựng các hàm lượng giá hành động theo công thức (1) cho chương trình VN-Connect, sau đó cho chương trình VN-Connect đấu với X6 và nhận được kết quả như trong Bảng 2. Kết quả cho thấy chương trình càng mạnh nếu đặc trưng càng tốt như trong bảng 1.

Bảng 1. So sánh tỷ lệ thắng thua giữa VN-Connect và X6

	MLE tổng quát	Kết quả	Tỷ lệ thắng thua VN–Connect – X6 (%)
4 mẫu 6	–3,0660602	176/1000	17,6
4 mẫu 7	–2,9578211	211/1000	21,10
2 mẫu 8, 2 mẫu 7	–2,8943371	273/1000	27,30

Bảng 2. Thay đổi thời gian suy nghĩ

Thời gian suy nghĩ (s)	Kết quả	Tỷ lệ thắng thua VN-Connect – X6 (%)
4	273/1000	27,30
6	482/1000	48,20
10	617,5/1000	61,75

Một thí nghiệm khác là so sánh thời gian suy nghĩ trong chương trình Monte Carlo. Thời gian suy nghĩ trong chương trình Monte Carlo rất quan trọng vì suy nghĩ càng nhiều thì việc lấy mẫu càng chính xác (số lượng playouts nhiều), dẫn đến việc hội tụ đến kết quả tối ưu. Chúng tôi thí nghiệm trên 6 giây và 10 giây và thấy kết quả khác biệt rõ rệt. Điều đó chứng tỏ kết quả thí nghiệm của việc chọn đặc trưng là chắc chắn và kết quả như dự đoán (Bảng 2).

7 Kết luận

Trong bài báo này, chúng tôi nghiên cứu các thành phần cơ bản của chương trình đánh cờ và xác định cụ thể cho từng loại cờ. Chương trình chúng tôi tìm hiểu và áp dụng là trò chơi Connect6. Đây là trò chơi có độ phức tạp tương đương cờ Vây. Cơ chế hoạt động của cây tìm kiếm Monte Carlo phù hợp cho những trò chơi có độ phân nhánh cao đã được xác định và hàm lượng giá hành động từ việc chọn ra các đặc trưng cũng được xây dựng. Các đặc trưng được rút trích từ các ván cờ có chất lượng và được xác định trọng số theo mô hình Bradley-Terry.

Việc xác định đặc trưng bước đầu được thực hiện thủ công và có kết quả đáng khích lệ. Hướng phát triển tiếp theo của bài báo này là sử dụng các phương pháp tối ưu ngẫu nhiên như Leo đồi, Luyện thép và Giải thuật di truyền; và sử dụng phương pháp Deep Learning để chọn lựa đặc trưng một cách tự động và tìm ra các đặc trưng tối ưu nhất, dẫn đến hàm lượng giá sẽ tối ưu.

TÀI LIỆU THAM KHẢO

1. XU Chang-ming, Z.M.MA; Yu Chang-yong;, XU Xin-he (2013), *A Pattern Based Incremental Model in K-in-a-row Games*, P.939–944.
2. Qiang Gao; Xinhe Xu (2016), *A Solving Strategy of Connect6 Based on K-in-a-row Types*, IEEE, P.5041–5045.
3. Amit Kumar Saxena; Vimal Kumar Dubey (2015), *A Survey on feature selection algorithms*, ISSN: 2321–8169, P. 1895–1899.
4. Francisco de Asis Boldt; Thomas W. Rauber and Flávio M. Varejão (2015), *Single sequence fast feature selection for high-dimensional data*, IEEE, P. 697–704.
5. Jung-Kuei Yang; Ping-Jung Tseng (2016), *Building connect6 Opening by using the Monte Carlo tree search*, IEEE, P. 331–336.

6. Huy Nguyen; Kokolo Ikeda; Simon Viennot (2014), *Fast Optimization of the Pattern Shapes in Board Games with Simulated Annealing*, Proceedings of the Sixth International Conference KSE 2014, pp. 325 – 337.
7. Huang, S.-C (2011), *New Heuristics for Monte Carlo Tree Search Applied to the Game of Go*, PhD Thesis, National Taiwan Normal University, Taipei, Taiwan, R.O.C.
8. Silver, D. and Tesauro, G. (2009), *Monte-Carlo simulation balancing*, In A. Danyluk, L. Bottou, and M. Littman, editors, ICML, ACM, volume 382, P. 945–952.
9. Loos, A. (2012), *Machine Learning for k-in-a-row Type Games Using Random Forest and Genetic Algorithm*, Master's thesis, University of Tartu, Tartu.
10. S.-J. Yen and J.-K. Yang (2011), *Two-Stage Monte Carlo Tree Search for Connect6*, IEEE Transactions on Computational Intelligence and AI in Games, 3 , pp.100–118.
11. S.-J. Yen, and J.-K. Yang (2010), *New Simulation Strategy of MCTS for Connect6*, the 15th Game Programming Workshop (GPW-2010), Hakone Seminar House, Kanagawa, Japan. GPW-2010 Proceeding pp. 90–93.

EXAMINATION AND EVALUATION OF FEATURE-SELECTIVE APPROACHES IN BOARD GAMES WITH HIGH-COMPLEXITY BRANCHES

Dang Cong Quoc[1], Nguyen Dang Binh[1], Nguyen Quoc Huy[2]

[1]Hue University of Science, 77 Nguyen Hue street, Hue city
[2] Sai Gon University, 273 An Duong Vuong street, district 5, HCM city

Abstract. Feature selection plays a crucial role in machine learning problem. Board game is a suitable testbed for AI areas, this is a really big challenge if the game with high complexity of branches like Go, Amazon, Connect6. It is very hard to find out the great features from game records in these games. This paper proposes a full survey of many studies in computer games such as search trees, evaluation functions, feature selection, and how to weight the game feature based on a set of game records.

Keywords: Feature selection, Connect6, evaluation function