Hàm lượng giá cho các thuật toán Minimax/AlphaBeta/Expectimax

Tên: Huỳnh Võ Ngọc Thanh

MSSV: 21520449

Lớp: CS106.O21.KHCL

-o-o-o-o-o-o-o-

Yêu cầu:

- Mô tả ý tưởng/chiến thuật tự thiết kế evaluation function. Những đặc trưng sử dụng để ước lượng giá trị trạng thái là gì? Trọng số của các đặc trưng đó như thế nào? Tại sao?
- Chạy thử nghiệm các thuật toán đã cài đặt với evaluation function đã thiết kế (so sánh với evaluation function có sẵn scoreEvaluationFunction trong đó chỉ sử dụng điểm số của trạng thái để ước lượng giá trị trạng thái). Hiệu năng/Hiểu quả của Minimax và AlphaBeta và Expectimax so với nhau thế nào?
- Thực nghiệm trên ít nhất 05 map khác nhau (trong thư mục layouts), mỗi map cần thực nghiệm ít nhất 05 lần (mỗi lần với 01 random seed khác nhau). Cụ thể:

  Đối với scoreEvaluationFunction, chạy Minimax, AlphaBeta, và Expectimax trên 05 map khác nhau (mỗi map cần chạy 05 lần với random seed là MSSV+0 -> MSSV+4).

  Đối với betterEvaluationFunction, chạy Minimax, AlphaBeta, và Expectimax trên 05 map đã chọn ở trên (mỗi map cần chạy 05 lần với random seed là MSSV+0 -> MSSV+4).

1 Mô tả ý tưởng/chiến thuật tự thiết kế evaluation function

Thiết kế một hàm lượng giá cho một trò chơi là yếu tố then chốt để trí tuệ nhân tạo (AI) có thể chơi trò chơi một cách hiệu quả và mang lại trải nghiệm tốt hơn cho người chơi. Mục tiêu của hàm lượng giá là ước tính giá trị của một trạng thái trò chơi tại một thời điểm cụ thể, giúp AI đưa ra quyết định tối ưu trong việc chọn hướng di chuyển và hành động trong trò chơi.

Để ước tính giá trị của một trạng thái, điểm số là một tính năng cần thiết vì nó phản ánh mức độ thành công hiện tại của trò chơi. Tuy nhiên, chỉ dựa vào điểm số để đánh giá trò chơi có thể không cung cấp thông tin chi tiết về các yếu tố cụ thể của trò chơi và có thể dẫn đến các quyết định kém tối ưu.

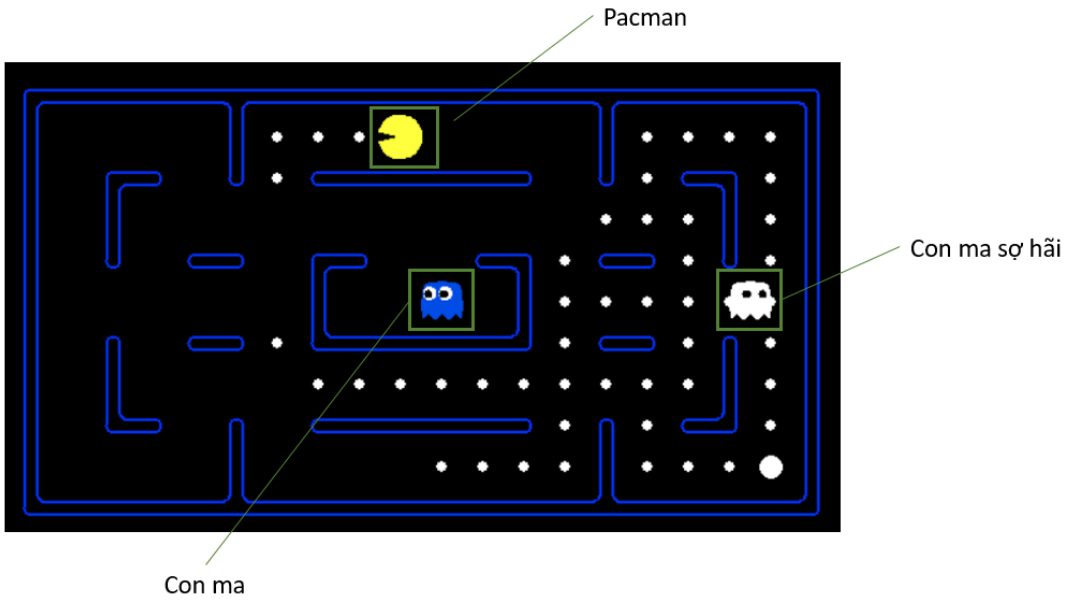

Figure 1: Hình ảnh minh họa cho game Pacman

Để thiết kế một hàm lượng giá tốt hơn, nên xem xét sử dụng nhiều đặc trưng khác nhau và trọng số tương ứng của chúng. Điều này cho phép AI đánh giá trạng thái hiện tại của trò chơi một cách chi tiết và toàn diện hơn, dẫn đến việc ra quyết định tốt hơn.

Hàm lượng giá **new_betterEvaluationFunction** trong trò chơi Pacman được thiết kế để tính toán giá trị của trạng thái trò chơi dựa trên nhiều yếu tố khác nhau nhằm hướng dẫn trí tuệ nhân tạo (AI) đưa ra quyết định tốt nhất cho Pacman.

Hàm lượng giá bắt đầu bằng việc lấy thông tin về trạng thái hiện tại của trò chơi, bao gồm vị trí của Pacman, danh sách chấm thức ăn và chấm thức ăn lớn, cùng với trạng thái của các con ma. Nó kiểm tra xem trò chơi có ở trạng thái thắng hoặc thua hay không và trả về giá trị tương ứng.

Sau đó, hàm lượng giá tiếp tục tính toán khoảng cách từ Pacman đến chấm thức ăn gần nhất, số lượng chấm lớn và thức ăn còn lại trong mê cung, cũng như phân loại ma thành ma hoạt động và ma đang sợ hãi. Nó tính toán khoảng cách đến ma hoạt động và ma đang sợ hãi, sau đó sử dụng các trọng số để tính điểm đánh giá của trạng thái hiện tại. Cuối cùng, hàm trả về điểm đánh giá để giúp AI đưa ra quyết định tối ưu trong trò chơi.

2 Các đặc trưng đã sử dụng và trọng số của chúng trong hàm lượng giá

Trong hàm lượng giá tự thiết kế, em đã sử dụng các đặc trưng để ước tính giá trị trạng thái của trò chơi:

currentScore là điểm số hiện tại của trạng thái. Sử dụng trọng số dương là 1 để khuyến khích Pacman đạt được điểm số cao hơn.

distanceToClosestFood là khoảng cách đến chấm thức ăn gần nhất. Trọng số âm 1.5 được sử dụng để giảm giá trị khi khoảng cách này giảm. Sử dụng trọng số âm 1.5 vẫn đủ để khuyến khích Pacman ưu tiên chọn các hành động đưa nó đến gần thức ăn, nhưng không gây quá nhiều áp lực khiến Pacman có thể bỏ qua các yếu tố khác trong trò chơi.

distanceToClosestActiveGhost là khoảng cách đến con ma hoạt động gần nhất. Sử dụng trọng số âm 2 và lấy nghịch đảo của khoảng cách để tạo ra giá trị lớn hơn khi Pacman ở xa con ma. Điều này khuyến khích Pacman tránh xa các con ma.

distanceToClosestScaredGhost là khoảng cách đến con ma sợ hãi gần nhất. Sử dụng trọng số âm 2 khuyến khích Pacman tiếp cận các con ma sợ hãi để ăn chúng. Trọng số này cao hơn trọng số của chấm thức ăn gần nhất do việc ăn các con ma sợ hãi không chỉ mang lại nhiều điểm số hơn là ăn thức ăn mà còn làm cho trò chơi dễ dàng hơn.

numberOfCapsulesLeft là số lượng chấm lớn còn lại trong trạng thái hiện tại. Sử dụng trọng số âm 20 để khuyến khích Pacman ăn chấm lớn khi đi qua chúng. Ở đây không ưu tiên ăn chấm thức ăn lớn hoặc tránh ma, nhưng Pacman nên ăn chấm lớn khi đi qua để tăng khả năng tiêu diệt ma và cũng thu được một số điểm từ việc ăn chấm lớn.

numberOfFoodsLeft là số lượng chấm thức ăn còn lại ở trạng thái hiện tại. Sử dụng trọng số âm 4 để đánh giá tiêu cực khi vẫn còn nhiều thức ăn. Điều này khuyến khích Pacman nên ăn hết thức ăn để hoàn thành màn chơi.

3 Kết quả thực nghiệm và rút ra kết luận

3.1 Thống kê kết quả thực nghiệm

Mỗi map em thực nghiệm 5 lần với random seed lần lượt là 215204490, 215204491, 215204492, 215204493, 215204494.

Hai maps originalClassic và powerClassic được thực thi với số bước nhìn trước là 2, contestClassic với số bước nhìn trước là 3. Còn lại được thực thi với số bước nhìn trước là 4.

Các score xanh dương cho biết màn chơi đó thắng, ngược lại score đỏ là màn chơi thua. Sau khi thực nghiệm, kết quả được thể hiện qua bảng 1 và bảng 2:

Maps	Minimax	Minimax + AlphaBeta	Expectimax
mediumClassic	Scores: 1149.0 -1417.0 -211.0 950.0 648.0 Average_Score: 223.8 Win_Rate: 2/5 (0.40)	Scores: 1149.0 -1417.0 -211.0 950.0 648.0 Average_Score: 223.8 Win_Rate: 2/5 (0.40)	Scores: 1149.0 -5210.0 -321.0 188.0 1280.0 Average_Score: -582.8 Win_Rate: 3/5 (0.60)
smallClassic	Scores: 364.0 1061.0 1311.0 1530.0 1089.0 Average_Score: 1071.0 Win_Rate: 4/5 (0.80)	Scores: 364.0 1061.0 1311.0 1530.0 1089.0 Average_Score: 1071.0 Win_Rate: 4/5 (0.80)	Scores: 1242.0 947.0 260.0 1541.0 1719.0 Average_Score: 1141.8 Win_Rate: 4/5 (0.80)
trappedClassic	Scores: -501.0 -501.0 -501.0 -501.0 -501.0 Average_Score: -501.0 Win_Rate: 0/5 (0.00)	Scores: -501.0 -501.0 -501.0 -501.0 -501.0 Average_Score: -501.0 Win_Rate: 0/5 (0.00)	Scores: 532.0 532.0 532.0 532.0 532.0 Average_Score: 532.0 Win_Rate: 5/5 (1.00)
contestClassic	Score: 52.0 -268.0 -459.0 452.0 -292.0 Average_Score: -103.0 Win_Rate: 0/5 (0.00)	Score: 52.0 -268.0 -459.0 452.0 -292.0 Average_Score: -103.0 Win_Rate: 0/5 (0.00)	Scores: -188.0 -268.0 1638.0 2606.0 749.0 Average_Score: 907.4 Win_Rate: 1/5 (0.20)
testClassic	Scores: 538.0 524.0 534.0 490.0 554.0 Average_Score: 528.0 Win_Rate: 5/5 (1.00)	Scores: 538.0 524.0 534.0 490.0 554.0 Average_Score: 528.0 Win_Rate: 5/5 (1.00)	Scores: 538.0 524.0 534.0 564.0 560.0 Average_Score: 544.0 Win_Rate: 5/5 (1.00)
originalClassic	Scores: -211.0 1224.0 566.0 124.0 -61.0 Average_Score: 328.4 Win_Rate: 0/5 (0.00)	Scores: -211.0 1224.0 566.0 124.0 -61.0 Average_Score: 328.4 Win_Rate: 0/5 (0.00)	Scores: -211.0 1224.0 496.0 -321.0 92.0 Average_Score: 256.0 Win_Rate: 0/5 (0.00)
powerClassic	Score: 203.0 183.0 1448.0 1021.0 631.0 Average_Score: 697.2 Win_Rate: 0/5 (0.00)	Score: 203.0 183.0 1448.0 1021.0 631.0 Average_Score: 697.2 Win_Rate: 0/5 (0.00)	Scores: 3.0 1189.0 -197.0 963.0 325.0 Average_Score: 456.6 Win_Rate: 0/5 (0.00)

Table 1: scoreEvaluationFunction

Maps	Minimax	Minimax + AlphaBeta	Expectimax
mediumClassic	Scores: 1920.0 1494.0 1640.0 2046.0 1846.0 Average_Score: 1789.2 Win_Rate: 5/5 (1.00)	Scores: 1920.0 1494.0 1640.0 2046.0 1846.0 Average_Score: 1789.2 Win_Rate: 5/5 (1.00)	Scores: 2091.0 1680.0 2016.0 2045.0 1321.0 Average_Score: 1830.6 Win_Rate: 5/5 (1.00)
smallClassic	Score: 1729.0 1504.0 1682.0 1729.0 960.0 Average_Score: 1520.8 Win_Rate: 5/5 (1.00)	Score: 1729.0 1504.0 1682.0 1729.0 960.0 Average_Score: 1520.8 Win_Rate: 5/5 (1.00)	Scores:1360.0 -163.0 1253.0 1544.0 1760.0 Average_Score: 1150.8 Win_Rate: 4/5 (0.80)
trappedClassic	Scores: 531.0 531.0 531.0 531.0 531.0 Average_Score: 531.0 Win_Rate: 5/5 (1.00)	Scores: 531.0 531.0 531.0 531.0 531.0 Average_Score: 531.0 Win_Rate: 5/5 (1.00)	Scores: 531.0 531.0 531.0 531.0 531.0 Average_Score: 531.0 Win_Rate: 5/5 (1.00)
contestClassic	Score: 1035.0 2662.0 2468.0 -48.0 2038.0 Average_Score: 1631.0 Win_Rate: 3/5 (0.60)	Score: 1035.0 2662.0 2468.0 -48.0 2038.0 Average_Score: 1631.0 Win_Rate: 3/5 (0.60)	Scores: 2879.0 2647.0 2266.0 884.0 2817.0 Average_Score: 2298.6 Win_Rate: 4/5 (0.80)
testClassic	Score: 520.0 518.0 550.0 494.0 526.0 Average_Score: 521.6 Win_Rate: 5/5 (1.00)	Score: 520.0 518.0 550.0 494.0 526.0 Average_Score: 521.6 Win_Rate: 5/5 (1.00)	Scores: 520.0 438.0 -62.0 494.0 496.0 Average_Score: 377.2 Win_Rate: 5/5 (1.00)
originalClassic	Scores: 1395.0 407.0 1331.0 2245.0 1774.0 Average_Score: 1430.4 Win_Rate: 1/5 (0.20)	Scores: 1395.0 407.0 1331.0 2245.0 1774.0 Average_Score: 1430.4 Win_Rate: 1/5 (0.20)	Scores: 3538.0 -71.0 2551.0 2765.0 1417.0 Average_Score: 2040.0 Win_Rate: 2/5 (0.40)
powerClassic	Score: 2074.0 3000.0 3285.0 2690.0 2267.0 Average_Score: 2663.2 Win_Rate: 3/5 (0.60)	Score: 2074.0 3000.0 3285.0 2690.0 2267.0 Average_Score: 2663.2 Win_Rate: 3/5 (0.60)	Scores: 3546.0 4729.0 3296.0 1916.0 2051.0 Average_Score: 3107.6 Win_Rate: 2/5 (0.40)

Table 2: new_betterEvaluationFunction

3.2 Nhận xét - Kết luận

3.2.1 So sánh new_betterEvaluationFunction - scoreEvaluationFunction

Dựa trên việc chạy thực nghiệm hàm lượng giá **new_betterEvaluationFunction** và hàm lượng giá có sẵn **scoreEvaluationFunction**, ta có nhận xét:

Hàm new_betterEvaluationFunction có tỷ lệ thắng cao hơn so với hàm scoreEvaluationFunction ở cả 3 thuật toán. Trong một số màn chơi như trappedClassic, contestClassic, powerClassic, hàm scoreEvaluationFunction gần như không thể chiến thắng, trong khi hàm new_betterEvaluationFunction có tỷ lệ thắng tăng lên đáng kể. Bên cạnh đó, hàm new_betterEvaluationFunction cũng đạt được chiến thắng trong đa số các lần thử nghiệm hơn hàm scoreEvaluationFunction ở 2 maps mediumClassic và smallClassic.

Hàm new_betterEvaluationFunction cũng cho điểm số trung bình cao

hơn so với hàm scoreEvaluationFunction ở hầu hết các màn chơi của cả 3 thuật toán.

Nhìn chung, khi bổ sung thêm 5 đặc trưng (distanceToClosestFood, distanceToClosestActiveGhost, distanceToClosestScaredGhost, numberOfCapsulesLeft, numberOfFoodsLeft) vào hàm lượng giá mới giúp cải thiện đáng kể khả năng giành chiến thắng và đạt được điểm số cao hơn của Pacman. Việc chỉ sử dụng điểm số của trạng thái để ước lượng giá trị trạng thái là chưa tối ưu.

3.2.2 So sánh hiệu quả của Minimax, Minimax-AlphaBeta, Expectimax

Minimax	**Minimax + AlphaBeta**	**Expectimax**
Average_Score: 320.6 Win_Rate: 11/35 (0.31)	Average_Score: 320.6 Win_Rate: 11/35 (0.31)	Average_Score: 465.0 Win_Rate: 18/35 (0.51)

Table 3: Điểm trung bình và tỷ lệ thắng ở 3 thuật toán với scoreEvaluationFunction

Minimax	**Minimax + AlphaBeta**	**Expectimax**
Average_Score: 1441.0 Win_Rate: 27/35 (0.77)	Average_Score: 1441.0 Win_Rate: 27/35 (0.77)	Average_Score: 1619.4 Win_Rate: 27/35 (0.77)

Table 4: Điểm trung bình và tỷ lệ thắng ở 3 thuật toán với new_betterEvaluationFunction

Dựa trên bảng 3 và bảng 4 kết hợp với quan sát kết quả thực nghiệm, thuật toán Expectimax dường như hiệu quả hơn so với hai thuật toán còn lại.

Trong hầu hết các thực nghiệm, Expectimax thường cho điểm số cao hơn so với Minimax và AlphaBeta Pruning. Điều này có thể là do Expectimax có khả năng xem xét tất cả các hành động có thể của tất cả các đối tượng. Hơn nữa, Expectimax tốt trong việc xử lý những mô hình có tính chất không chắc chắn. Điều này phù hợp trong Pacman - môi trường có tính chất không chắc chắn khi hành động của các con ma không được biết trước một cách chính xác.

Tuy nhiên nếu xét trên từng map cụ thể (smallClassic, testClassic) thì vẫn tồn tại trường hợp Expectimax không tối ưu bằng hai thuật toán trên do việc ước lượng theo giá trị kì vọng có thể chưa phù hợp trong một layout cụ thể của Pacman.

Hai thuật toán Minimax và AlphaBeta có kết quả thực nghiệm tương tự nhau do hai thuật toán này đều dựa trên ý tưởng cơ bản của thuật toán Minimax, nhưng thời gian xử lý của AlphaBeta sẽ tốt hơn nhiều so với Minimax do cơ chế pruning cắt tỉa các nhánh không cần thiết, AlphaBeta giúp giảm

thiểu thời gian tính toán mà không làm ảnh hưởng đến kết quả cuối cùng của trò chơi.

Các thực nghiệm ở đây chỉ trong một phạm vi nhất định với số lượng testcase tương đối nhỏ và có thể bị ảnh hưởng bởi các yếu tố bên ngoài, do đó các kết luận có thể mang tính chủ quan. Việc chọn trọng số cho các đặc trưng trong new_betterEvaluationFunction cũng mang tính chủ quan, tức là vẫn tồn tại một bộ trọng số khác cho hiệu quả tốt hơn.

Link drive record lại màn chơi mà em thấy cho kết quả tốt nhất (4729.0):

https://drive.google.com/file/d/1SS-upYJdmrCdgOxAukjLINww0IUS7KX6/view?usp=sharing