ĐẠI HỌC QUỐC GIA TP.HCM
TRƯỜNG ĐẠI HỌC CÔNG NGHỆ THÔNG TIN

KHOA: KHOA HỌC MÁY TÍNH

MÔN: TRÍ TUỆ NHÂN TẠO - CS106.O21.KHTN

Evaluation functions for MultiAgents

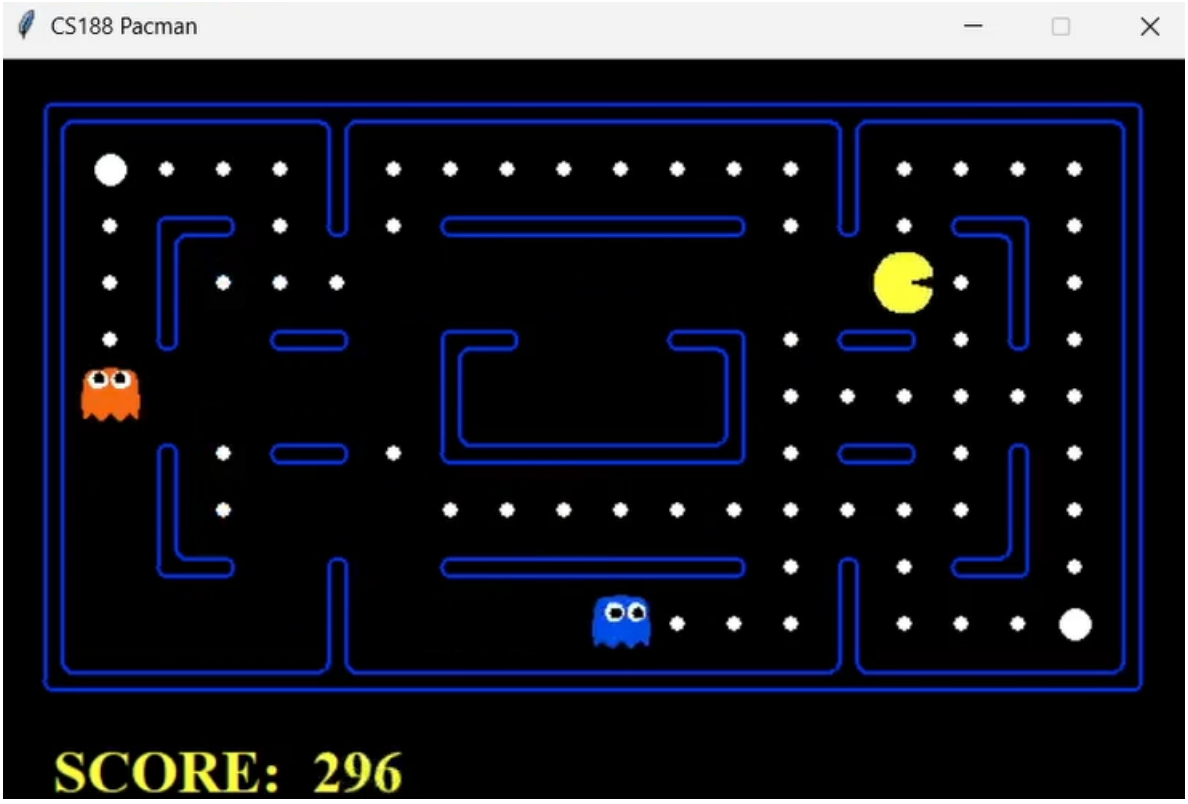
Sinh viên thực hiện:
Trương Thanh Minh - 21520064

Mục lục

1 Giới thiệu tổng quan

1.1 Giới thiệu game Pacman

Pacman là một trò chơi gồm 3 thành phần chính: Pacman, Ghost và Food. Trong đó Pacman là hình tròn bị khuyết màu vàng. Còn Ghost là các con màu cam và xanh trong hình 1, trong game có thể có nhiều Ghost, nhưng mà chỉ có một Pacman mà thôi. Các chấm bé bé màu trắng gọi là Food. Nhiệm vụ của Pacman là phải ăn hết các chấm thức ăn càng nhanh càng tốt và không được chạm và Ghost. Tuy nhiên Pacman có thể ăn được Ghost nếu trước đó Pacman ăn chấm thức ăn màu trắng to to như trong ảnh (khi đó, tất cả các ghost sẽ biến thành màu trắng). Tuy nhiên, điều đó sẽ hết hiệu lực trong 1 khoảng thời gian cố định (rất ngắn). Vì score sẽ giảm dần nếu Pacman đứng yên hoặc không ăn được chấm thức ăn nào. Vì vậy, nhiệm vụ của ta là phải giúp Pacman ăn hết các chấm thức ăn nhanh nhất có thể và phải tuân theo quy tắc của trò chơi.



Hình 1: Giao diện game Pacman

1.2 Yêu cầu

Thực hành trong file *multiAgents.py* với các nội dung sau:

- Minimax (trong source code tham khảo đính kèm đã cài đặt 2 phiên bản).
- Minimax với Alpha-Beta pruning.

- Expectimax (xem như các ghost agents lựa chọn hành động với xác suất ngẫu nhiên đồng đều).
- Tự thiết kế hàm lượng giá (evaluation function) để ước lượng giá trị của các trạng thái trung gian (trong source code tham khảo đính kèm đã có sẵn hàm betterEvaluationFunction gợi ý).

1.3 Source code

Link github: https://github.com/trthminh/CS106.O21.KHTN/tree/main/multiagent.

2 Evaluation functions

Trong hàm tự lượng giá mà em tự thiết kế sử dụng các thông tin sau:

1. *currentGameState.getScore()*: Score của trạng thái hiện tại. Đặt là F1.
2. *ghost_score*: Để tính được thông tin này, dựa vào khoảng cách euclide giữa Pacman và ghost. Với mỗi ghost, ta sẽ kiểm tra xem liệu ghost đó có đang bị scared hay không (tức là Pacman có thể ăn ghost đó). Nếu $scaredTimer$ của ghost > 0, tức là Pacman có thể ăn được ghost đó, khi đó ta sẽ tăng giá trị ghost_score lên một khoảng $100/distance$. Còn ngược lại ta sẽ giảm giá trị này xuống $10/distance$. Đặt là F2
3. *pacmanDirectsScore*: thông tin này thể hiện cho các hướng mà Pacman có thể di chuyển trong state hiện tại. Ta khuyến khích Pacman ưu tiên trạng thái có nhiều hướng di chuyển hơn. Vì khi đó sẽ tăng cơ hội chiến thắng của Pacman hơn. Đặt là F3
4. *closestCapsule*: thông tin này thể hiện Capsule gần với Pacman nhất. Nếu tồn tại Capsule thì sẽ cập nhật lại giá trị đó bằng $-3/closestCapsule$. Ngược lại, nếu không tồn tại, thì ta sẽ cập nhật giá trị bằng 100. Ta thực hiện điều này với mong muốn khuyến khích Pacman có thể ăn Capsule để tăng cơ hội ăn ghost. Đặt là F4.
5. *closestFood*: thông tin này cho biết chấm thức ăn nào gần với Pacman nhất. Đặt là F5.
6. *foodList*: Danh sách các chấm thức ăn còn lại trong map. Đặt là F6

Cuối cùng, ta sẽ có được công thức sau:

$$final\_score = F1 + F2 + F3 + F4 - 4 * F5 - 10 * len(F6)$$

3 Thực nghiệm

Để chạy thực nghiệm, em thực hiện trên 5 map: mediumClassic, capsuleClassic, contestClassic, minimaxClassic và trappedClassic. Với mỗi map, em chạy 5 lần với random

seed từ 21520064 -> 21520068 và depth=3. Sau đó, em tính trung bình 5 lần chạy đó lại và ghi vào cột Scores, số lần thắng trong 5 lần chạy em ghi vào cột #Win.

Sau đó, em chạy Minimax, AlphaBeta và Expectimax giống như đề cập ở trên với 2 hàm đánh giá: scoreEvaluationFunction và MinhsbetterEvaluationFunction (hàm lượng giá em tự thiết kế).

	MinimaxAgent		AlphaBetaAgent		ExpectimaxAgent	
Layout	Scores	#Win	Scores	#Win	Scores	#Win
scoreEvaluationFunction						
mediumClassic	685.6	1	685.6	1	227.6	2
capsuleClassic	-304.2	0	-304.2	0	-71.6	0
contestClassic	170.8	0	170.8	0	772.4	0
minimaxClassic	-293.4	1	-293.4	1	-98.8	2
trappedClassic	-501	0	-501	0	325.2	4
MinhsbetterEvaluationFunction						
mediumClassic	1814.8	5	1814.8	5	578	5
capsuleClassic	46	0	46	0	433.2	0
contestClassic	3033.8	4	3033.8	4	2788	3
minimaxClassic	110.2	3	110.2	3	309	4
trappedClassic	-501	0	-501	0	324.4	4

Bảng 1: Bảng thực nghiệm kết quả với hai hàm đánh giá

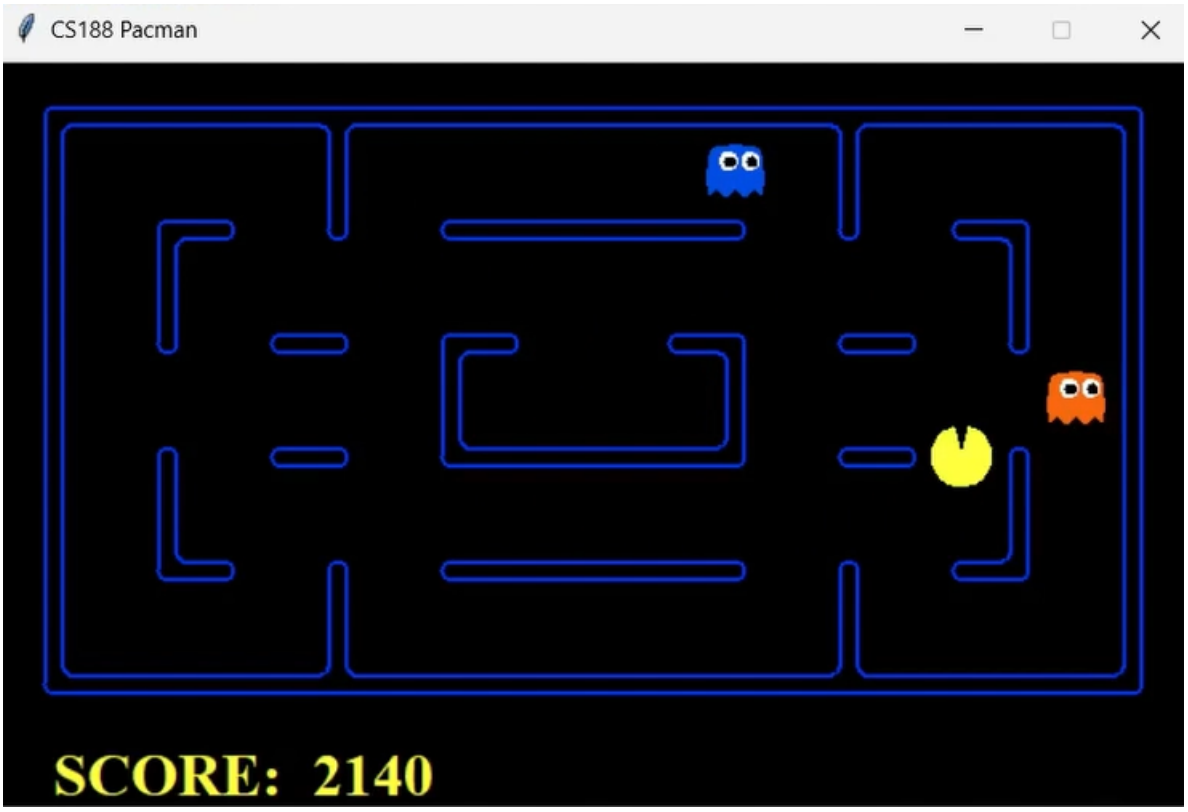
4 Nhận xét

Từ bảng kết quả trên, có thể thấy, hiệu quả của MinimaxAgent và AlphaBetaAgent là như nhau. Ngoài ra, hiệu quả của ExpectimaxAgent nhìn chung tốt hơn so với MinimaxAgent và AlphaBetaAgent.

Nhìn chung, **số bàn thắng** của ExpectimaxAgent so với MinimaxAgent và AlphaBetaAgent đều **cao hơn**, **scores** trung bình trong 5 lần chơi ở mỗi map cũng **tốt hơn.** Tuy nhiên, trong một số map như mediumClassic và contestClassic đôi khi score lại thấp hơn so với MinimaxAgent và AlphaBetaAgent.

Trong tất cả các lần thử nghiệm, kết quả cho ra khi dùng *MinhsbetterEvaluationFunction* đều tốt hơn so với *scoreEvaluationFunction*.

Trong các lần thử nghiệm, lần thử nghiệm mà em thấy có kết quả cao nhất là khi em thử trong **map mediumClassic với MinimaxAgent**, khi đó Pacman đã **win** với số điểm cao nhất là **2140**. Chi tiết trong link demo ở đây.



Hình 2: Hình ảnh Pacman với lần thử nghiệm tốt nhất