Evaluation Functions for Minimax/AlphaBeta/Expectimax Report

Le Thi Lien
Student ID : 21522282

(May 20, 2023)

1. Idea/Strategy

Designing an evaluation function for a game is an important part of building an effective artificial intelligence (AI) for the game. The goal of the evaluation function is to estimate the value of an intermediate state, helping the AI make optimal decisions in choosing actions.

To estimate the value of a state, the score is a necessary feature as it reflects the current level of success of the player. However, relying solely on the score as a feature in the evaluation function may have limitations. The score alone does not provide detailed information about specific elements of the game and can lead to suboptimal decisions.

Therefore, to design a better evaluation function, it is advisable to consider using multiple different features and their corresponding weights. This allows the AI to evaluate the current state of the game in a more detailed and holistic manner, leading to better decision-making.

2. Used Features & Weights

In the my evaluation function, I have used 6 features to estimate the value of the Pacman game state. Here is a detailed analysis of each feature and its corresponding weight:

(i) **currentScore**: This is the current score of the state. I use a positive weight of **1** to encourage Pacman to achieve a higher score.

(ii) **distanceToClosestFood**: This is the distance to the closest food. I use a weight of **-1.5** to decrease the value as this distance decreases, meaning Pacman gets closer to the food. This encourages Pacman to choose actions that bring it closer to the food.

(iii) **distanceToClosestActiveGhost**: This is the distance to the closest active ghost. I use a weight of **-2** and take the inverse of the distance to create a larger value when Pacman is farther away from the ghost. This encourages Pacman to stay away from the ghosts.

(iv) **distanceToClosestScaredGhost**: This is the distance to the closest scared ghost. I use a weight of **-2** to decrease the value as this distance decreases, meaning Pacman gets closer to the scared ghost. This encourages Pacman to approach scared ghosts to eat them. This weight is higher than the weight of the closest food (even though

the distance to the scared ghost is usually greater than the distance to the nearest food) because eating scared ghosts provides significantly more points than eating food.

(v) **numberOfCapsulesLeft**: This is the number of remaining capsules in the current state. I use a weight of **-20** to encourage Pacman to eat capsules when passing by them. I don't prioritize eating capsules over food or avoiding ghosts, but it's worth it for Pacman to eat capsules when passing by them to increase the ability to eliminate ghosts and also gain some points from eating the capsules.

(vi) **numberOfFoodsLeft**: This is the number of remaining food in the current state. I use a weight of **-4** to evaluate negatively when there is still a lot of food. This encourages Pacman to eat all the food to complete the level.

3. Experiments

3.1. *Comparison with scoreEvaluationFunction*

Based on the results of running experiments with the **'betterEvaluationFunction'** and the pre-existing **'scoreEvaluationFunction'**, we have gained a clear understanding of the effectiveness of the new evaluation function.

The **'betterEvaluationFunction'** has a significantly higher win rate compared to the **'scoreEvaluationFunction'**. In some game maps, such as capsuleClassic, the **'scoreEvaluationFunction'** is unable to achieve a win, while the **'betterEvaluation-Function'** performs well. Additionally, the **'betterEvaluationFunction'** outperforms the **'scoreEvaluationFunction'** in the mediumClassic game, consistently achieving wins in multiple trial runs.

Furthermore, the **'betterEvaluationFunction'** helps to achieve a much larger difference in average scores compared to using the **'scoreEvaluationFunction'**. In my experiments, this difference was observed to be ninefold.

Therefore, after incorporating 5 additional features into my evaluation function, it is evident that they greatly enhance Pacman's ability to win and achieve higher scores. Relying solely on the score feature is not sufficient; the inclusion of other features is crucial.

3.2. *Performance Comparison of Minimax, AlphaBeta, and Expectimax Algorithms*

Based on the experimental observations of the three implemented algorithms, Minimax, Alpha-Beta, and Expectimax, we have the following observations:

All three algorithms have approximately the same win rate (0.74, 0.77, 0.74) when using the **'betterEvaluationFunction'**.

However, there is a significant difference in the achieved scores among the three algorithms. For example, in the contestClassic map, Minimax has an average score of 1697.2, while Expectimax has a much higher score of 2409.6 and can even

Layout	Minimax	AlphaBeta	Expectimax
small	Average Score: 1250.2 Scores: 1119.0, 980.0, 937.0, 1697.0, 1518.0 Win Rate: 5/5 (1.00)	Average Score: 1147.8 Scores: 307.0, 1120.0, 1230.0, 1362.0, 1720.0 Win Rate: 4/5 (0.80)	Average Score: 1260.6 Scores: 298.0, 1521.0, 1370.0, 1744.0, 1370.0 Win Rate: 4/5 (0.80)
medium	Average Score: 1808.8 Scores: 1602.0, 1543.0, 1918.0, 1874.0, 2107.0 Win Rate: 5/5 (1.00)	Average Score: 1702.8 Scores: 2074.0, 1503.0, 1419.0, 1693.0, 1825.0 Win Rate: 5/5 (1.00)	Average Score: 1899.2 Scores: 2075.0, 2015.0, 1400.0, 2094.0, 1912.0 Win Rate: 5/5 (1.00)
contest	Average Score: 1697.2 Scores: 2545.0, 2024.0, 1894.0, -431.0, 2454.0 Win Rate: 4/5 (0.80)	Average Score: 2030.6 Scores: 1792.0, 2414.0, 1923.0, 3025.0, 999.0 Win Rate: 4/5 (0.80)	Average Score: 2409.6 Scores: 3638.0, 3631.0, 127.0, 2247.0, 2405.0 Win Rate: 4/5 (0.80)
minimax	Average Score: 104.8 Scores: 510.0, 511.0, 509.0, -503.0, -503.0 Win Rate: 3/5 (0.60)	Average Score: 107.2 Scores: -495.0, 511.0, 512.0, 511.0, -503.0 Win Rate: 3/5 (0.60)	Average Score: 103.4 Scores: -503.0, 507.0, 505.0, 511.0, -503.0 Win Rate: 3/5 (0.60)
capsule	Average Score: 401.6 Scores: 1443.0, 1234.0, -333.0, 161.0, -497.0 Win Rate: 2/5 (0.40)	Average Score: 622.4 Scores: 1211.0, -441.0, -440.0, 1373.0, 1409.0 Win Rate: 3/5 (0.60)	Average Score: 430.4 Scores: -456.0, 1175.0, 1947.0, -441.0, -73.0 Win Rate: 1/5 (0.20)
trapped	Average Score: -88.8 Scores: -502.0, 531.0, -502.0, 531.0, -502.0 Win Rate: 2/5 (0.40)	Average Score: 117.8 Scores: 531.0, 531.0, -502.0, -502.0, 531.0 Win Rate: 3/5 (0.60)	Average Score: 324.4 Scores: 531.0, 531.0, -502.0, 531.0, 531.0 Win Rate: 4/5 (0.80)
test	Average Score: 551.6 Scores: 538.0, 544.0, 562.0, 562.0, 552.0 Win Rate: 5/5 (1.00)	Average Score: 553.2 Scores: 562.0, 562.0, 562.0, 542.0, 538.0 Win Rate: 5/5 (1.00)	Average Score: 524.0 Scores: 550.0, 466.0, 518.0, 522.0, 564.0 Win Rate: 5/5 (1.00)

Figure 1 : 'betterEvaluationFunction'

Layout	Minimax	Alphabeta	expecttimax
small	Average Score: 713.6 Scores: 1278.0, 1284.0, -170.0, -7.0, 1183.0 Win Rate: 3/5 (0.60)	Average Score: 510.2 Scores: 1319.0, -183.0, 1103.0, 249.0, 63.0 Win Rate: 2/5 (0.40)	Average Score: 669.0 Scores: 714.0, 1137.0, -123.0, 1244.0, 373.0 Win Rate: 3/5 (0.60)
medium	Average Score: 362.4 Scores: -810.0, 408.0, 798.0, 1390.0, 26.0 Win Rate: 2/5 (0.40)	Average Score: -290.4 Scores: -1532.0, 275.0, 90.0, -351.0, 66.0 Win Rate: 1/5 (0.20)	Average Score: 532.0 Scores: -245.0, 736.0, 1587.0, 306.0, 276.0 Win Rate: 2/5 (0.40)
contest	Average Score: -245.8 Scores: -600.0, -246.0, -462.0, -364.0, 443.0 Win Rate: 0/5 (0.00)	Average Score: -138.6 Scores: -289.0, -138.0, -40.0, -30.0, -196.0 Win Rate: 0/5 (0.00)	Average Score: 765.0 Scores: -113.0, 728.0, 1860.0, 1577.0, -227.0 Win Rate: 1/5 (0.20)
minimax	Average Score: -92.2 Scores: -495.0, 510.0, 513.0, -492.0, -497.0 Win Rate: 2/5 (0.40)	Average Score: -293.8 Scores: -495.0, -498.0, -495.0, -492.0, 511.0 Win Rate: 1/5 (0.20)	Average Score: 309.2 Scores: 512.0, 507.0, -497.0, 511.0, 513.0 Win Rate: 4/5 (0.80)
capsule	Average Score: -428.8 Scores: -378.0, -532.0, -510.0, -568.0, -156.0 Win Rate: 0/5 (0.00)	Average Score: -475.2 Scores: -512.0, -512.0, -433.0, -473.0, -446.0 Win Rate: 0/5 (0.00)	Average Score: -13.0 Scores: 105.0, -300.0, 113.0, 125.0, -108.0 Win Rate: 0/5 (0.00)
trapped	Average Score: -501.0 Scores: -501.0, -501.0, -501.0, -501.0, -501.0 Win Rate: 0/5 (0.00)	Average Score: -501.0 Scores: -501.0, -501.0, -501.0, -501.0, -501.0 Win Rate: 0/5 (0.00)	Average Score: -295.2 Scores: -502.0, -502.0, -502.0, -502.0, 532.0 Win Rate: 1/5 (0.20)
test	Average Score: 528.6 Scores: 514.0, 485.0, 530.0, 554.0, 560.0 Win Rate: 5/5 (1.00)	Average Score: 538.8 Scores: 512.0, 536.0, 558.0, 542.0, 546.0 Win Rate: 5/5 (1.00)	Average Score: 548.8 Scores: 506.0, 564.0, 564.0, 558.0, 552.0 Win Rate: 5/5 (1.00)

Figure 2 : 'scoreEvaluationFunction'

layout	Minimax	Alphabeta	Expectimax	
betterEvaluationFunction	817.9	897.4	993.1	902.8
scoreEvaluationFunction	48.1	-92.9	359.4	104.5
	433	402.3	676.3	

reach over 3600 points. Based on the average scores of the three algorithms using **'betterEvaluationFunction'** in my experiments, Expectimax appears to be much more effective than the other two algorithms.

4. Conclusion

However, it is important to note that my experiments had a relatively small number of trials and were also influenced by various external factors, so the conclusions drawn are still subjective. The choice of weights for the features in the evaluation function also relied on my own intuition and could be further improved to achieve the best results.

Link to the game with the highest score: `https://drive.google.com/drive/folders/1HTgM6DlBS-ITByFrBspN7FqIOF-B6PUU?usp=sharing`