Advanced Artificial Intelligence 2 CMP9132M

James Tombling (25913882)

February 2022

1 Aims and Objects

The aim of this project was to build a code artefact for navigating the "meanArena" for as long as possible while collecting the "Bonuses" and avoiding the static "pits" and the moving "meanies". By deploying artificial intelligence (AI) techniques this can be done to varying degrees of success. The success of the project can be evaluated from how well the player ("Tallon") can perform in different sized arenas with varying numbers of pits, bonuses and meanie spawning times.

2 Literature Review

Three types of learning paradigms were considered for this project, these include; Markov Decision Process (MDP), deep Reinforcement Learning and a class of evolutionary methods, called Genetic Algorithms. Deep Reinforcement learning is a combination of reinforcement learning and deep neural networks, it works on the basis that the AI retains information about prior experiences and the results gained so it can make a decision next time about which solution is best to take and the more iterations the network is given to converge the better the AI tends to perform, thus potentially being the best solution for game solving AI. however this process can be take a long time to converge and requires a large amount of computational power. There is also no guarantee that it will work in the desired way. (Li, 2017).

A genetic algorithm implements the concept of computation by representing the chromosomes with arrays of bits or characters (binary string). Each string corresponds to a possible solution. The genetic algorithm then tweaks the most promising chromosomes in pursuit of better results. these chromosomes could be the potential action of "Tallon" given a specific state and would be less computationally heavy than deep reinforcement Learning.(Henderson et al., 2018)

A Markov decision process (MDP) is a discrete-time stochastic control process in mathematics. It gives a mathematical framework for modelling decision-making in settings where outcomes are partially random and partly controlled by a decision maker. Thus meaning this methods fits perfectly with the non-deterministic features of the "meanArena" game. MDPs can be used to investigate optimization problems that are solved using dynamic programming, such as getting to the "bonuses" while avoiding "pits" and "meanies". MDPs have been recognised a an effective way to solve these kinds of problems since Ronald Howard's 1960 book, "Dynamic Programming and Markov Processes",which sparked a large corpus of study on Markov decision processes. (Howard, 1960). Robotics, automatic control, economics, and manufacturing are just a few of the fields where they're applied. MDPs are an extension of Markov chains and are named after Russian mathematician Andrey Markov. MDPs pose to be a more elegant and less computationally demanding method solving the "meanArena" game than using deep Reinforcement Learning AI algorithms under the umbrella of MDPs which could be used in this project include; Q-learning, Policy Iteration and Value Iteration. (Even-Dar et al., 2004).

in Q-Learning the point is that the agent does not know the probabilities or rewards of state transitions. When the agent goes from one state to another by a certain activity and earns a reward,and only then will it realises that there is a reward. Similarly, it can only determine what transitions are accessible from a given state by arriving there and examining the alternatives. It learns the likelihood of transitioning between states if state transitions are stochastic. It does this by observing how frequently distinct transitions occur.(Watkins and Dayan, 1992, 279). When you have transition probabilities, that is, when you know the likelihood of getting from state A to state B with action X, Value-Iteration is utilised. On the other hand, you might have a program that allows you to mimic it, but you don't get the likelihood. So you no longer have a model. This is when Q learning comes into play. Value iteration can also only return the policy of the next state A to state B move.

Value-Iteration includes finding the optimal value and getting one policy extraction. because if the value is optimal then the policy should be optimal as well, hence it does not repeat its itself and its deterministic outcome would not suite the non-deterministic nature of the "meanArena" game. (Dai and Goldsmith, 2007, 1860).

Policy-Iteration using the main process of the Value-Iteration and includes taking a policy evaluation at a given state and a policy improvement and unlike the Value-Iteration it repeats them until the policies converge. thus allowing for a environment which includes and model and is non-deterministic process to be utilised, unlike Q-Learning and Value-iteration. Thus making Policy-Iteration the most suitable non-network non-deterministic model based solution to this project and is the most appropriate solution to this problem. (Alla et al., 2015, A181).

The algorithm for Policy Iteration can be seen in figure 1 below, this is what the code for this project will be based upon.

Figure 1 Policy-Iteration Algorithm

1. Initialization
   $V(s) \in \Re$ and $\pi(s) \in \mathcal{A}(s)$ arbitrarily for all $s \in \mathcal{S}$

2. Policy Evaluation
   Repeat
   $\Delta \leftarrow 0$
   For each $s \in \mathcal{S}$:
   $v \leftarrow V(s)$
   $V(s) \leftarrow \sum_{s'} \mathcal{P}_{ss'}^{\pi(s)} \left[\mathcal{R}_{ss'}^{\pi(s)} + \gamma V(s')\right]$
   $\Delta \leftarrow \max(\Delta, |v - V(s)|)$
   until $\Delta < \theta$ (a small positive number)

3. Policy Improvement
   policy-stable $\leftarrow$ *true*
   For each $s \in \mathcal{S}$:
   $b \leftarrow \pi(s)$
   $\pi(s) \leftarrow \arg\max_a \sum_{s'} \mathcal{P}_{ss'}^{a} \left[\mathcal{R}_{ss'}^{a} + \gamma V(s')\right]$
   If $b \neq \pi(s)$, then *policy-stable* $\leftarrow$ $false$
   If *policy-stable*, then stop; else go to 2

3 Methodology

The Method for creating the MDP Policy-Iteration includes a system for generating Reward matrices and Probability matrices. The probability matrix is a 4 by map-size by map-size where the map-size array is the X-axis by Y-axis of the arena. for the basic "meanArena" the X-axis is 10 and Y-axis is 10 making the map-size 100 this represents the 100 states in relation to the 100 other states of the arena and the 4 represents the number of possible directions "Tallon" can move(North, South, East and West). Thus the probability matrix represents all possible moves "Tallon" can take from any state concerning all other points in Arena with its forward move possibility being 95 percent and its side moves being 2.5 percent each. each 40,000 matrices must sum to 1. The probability matrix is filled through a series of For loops that iterates through the different directions for all the different states in the arena and the if statements filter through the corners, sides of the map then an else statement fill the rest.

The Reward matrix is a map-size by 4 matrix that contains 100 lots of 4 value long arrays. each of these 4 long arrays is a specific state in the map starting with 0 being the top left corner of the arena and 99 being the bottom right going along in rows. Each of these 4 long arrays is filled according to what is present at that state in the arena, if its a "bonus" all 4 of the values in that state specific array is filled with a positive value indicating the 4 directions surrounding that state (point in the arena). This process is repeated for the "pits" and "meanies" but with negative values as "Tallon" wants to avoid these. The reward matrix also fills all edge and corner states with negative values but not as negative as "pits" and "meanies" , this prevents "Tallon" getting cornered by the "meanies"

4 Evaluation

To Evaluate the performance a series of 10 tests was carried out for 11 different size arenas and an average of their score was generated.

	Runs										
size of arena	1	2	3	4	5	6	7	8	9	10	Average
5 x 5	0	26	12	11	12	11	13	13	10	22	13
6 x 6	15	17	24	27	11	12	26	15	14	24	18.5
7 x 7	14	29	23	28	12	24	14	15	16	18	19.3
8 x 8	13	15	27	24	32	33	28	26	33	15	24.6
9 x 9	14	38	25	30	25	25	24	25	26	27	25.9
10 x 10	29	31	33	33	20	40	6	35	24	41	29.2
11 x 11	44	25	37	35	26	32	21	29	41	21	31.1
12 x 12	29	35	28	38	39	38	36	38	39	38	35.8
13 x 13	35	37	36	38	39	37	35	37	38	34	36.6
14 x 14	27	39	38	37	39	41	27	42	39	41	37
15 x 15	35	38	42	35	36	42	39	36	38	39	38

Figure 2 Table of Average scores for different size arenas

Any data highlighted in the tables has been considered an error because the way the game works its has the potential to spawn "Tallon" on top off "pits" or spawn "meanies" on top off "Tallon" or in a very compromising position leading to low scores.

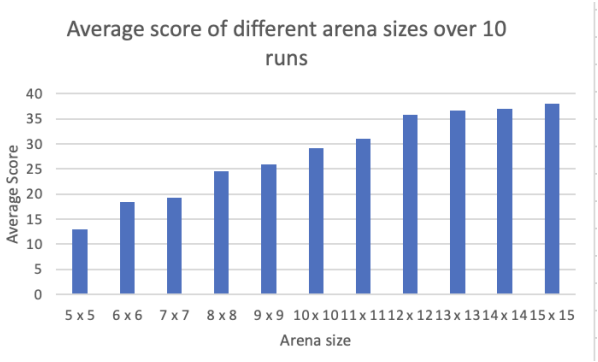

Figure 3 Average Score Graph for Arena Sizes

As seen in figure 2 and 3 as the size of the arena increases the average score does as well as "Tallon" has more space to avoid the "Pits" and "Meanies" but starts converging at 13 by 13 as at this point so many "Meanies" have spawned in "Tallon" cant escape them. The smaller size arenas have a lower score due to the arena limiting the potential pathways "Tallon" can take to avoid obstacles meaning it is caught by the "meanies" quicker or falls into a "Pit" due to mistakes.

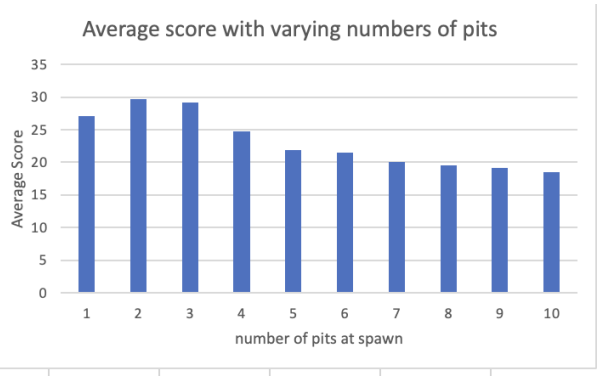

Figure 4 Average Score Graph for number of pits

Figure 4 shows the average score with varying numbers of "Pits" at the start of the game. All other features of the game such as the "meanie" spawn rate, number of "bonuses" and size of the arena are all kept the same as the original game configurations. With a lower number of "pits", the scoring average is higher as "Tallon" has more possible polices pathways to follow and is less likely to statistically fall into a pit

or get cornered and caught by a "Meanie". As The number of "pits" increases the average score decrease but starts to plateau around 18 this may be due to the fact that the "Pits" only make up a max of 10 percent of the map in this case, with a bigger arena it would require more "Pits" to decrease the average score. Find the table of data for figure 4 in appendix 1.

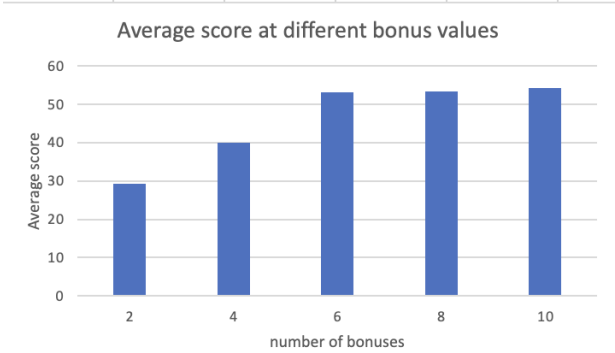

Figure 5 Average Score Graph for number of bonus

Figure 5 shows the average score with varying numbers of "bonuses" at the start of the game. All other features of the game such as the "meanie" spawn rate, number of "Pits" and size of the arena are all kept the same as the original game configurations. As the number of "Bonuses" increases the average score increases as well, which was expected as each "bonus" is worth 10 points so the minimum average for each number of "bonuses" should be the number of "bonuses" multiplied by 10. The test with 2 "Bonuses" exceeds this as "Tallon" can quickly collect the two "bonuses" and proceed to gain extra score by avoiding the "Meanies". The higher number of Bonuses tends to Plateau as by the time "Tallon" has tried to reach all of these "Bonuses" , as by this many iterations to navigate the map the number of "meanies" can start to overwhelm "Tallon". Thus the 2 "bonuses" averages just under 50 percent extra that anticipated and the 10 "bonuses" averages almost 50 percent less that anticipated. The corresponding table for this graph can be found in appendix 2.

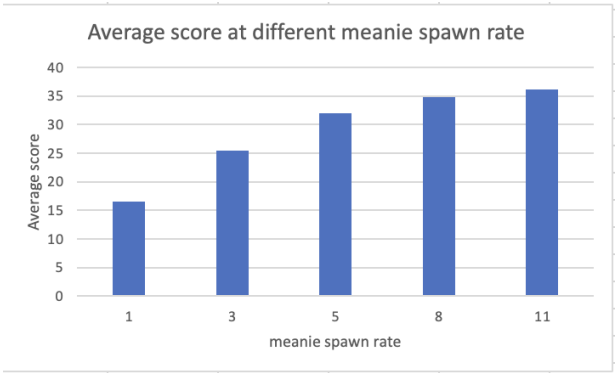

Figure 6 Average Score Graph for meanie spawn rate

Figure 6 shows the average score with varying spawn rates of the "Meanies" at the start of the game. All other features of the game such as the number of "bonuses", number of "Pits" and size of the arena are all kept the same as the original game configurations.

As expected when the "Meanie" spawn rate is at 1 a new "meanie" spawns every iteration which very quickly overwhelms the "Tallon" which mean a high score cannot be achieved. Thus the number for the meanie spawn rate has a positive correlation to the average score game as seen in figure 6.

In conclusion, the AI worked as expected however the average scores for the number of "Bonuses" and the number of "Pits" at the start could be improved by creating a buffer value around the "Pits" and "meanies" of a higher than -1 but less than 0 negative number such as -0.3 in the surrounding 8 squares of each enemy as seen in figure 7 below.

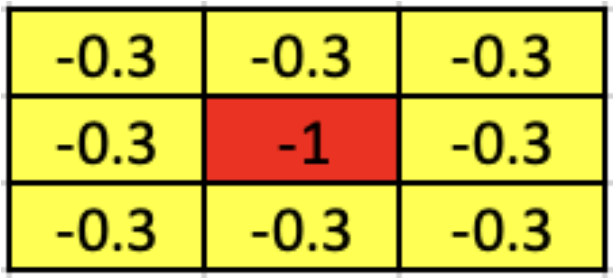

Figure 7 Negative Buffer

This new buffer zone would help the avoidance part of the AI algorithm and would be considered for further development of the algorithm. Another way to increase the effectiveness of the system could be to implement a Partially Observable Markov Decision Process (POMDP) due to the partial visibility part of the project and should be considered for further improvement.

5 References

Alla, A., Falcone, M. and Kalise, D. (2015) An efficient policy iteration algorithm for dynamic programming equations. SIAM Journal on Scientific Computing, 37 (1) A181-A200.

Dai, P. and Goldsmith, J. (2007) Topological Value Iteration Algorithm for Markov Decision Processes. In: Anonymous IJCAI 1860-1865.

Even-Dar, E., Kakade, S.M. and Mansour, Y. (2004) Experts in a Markov decision process. Advances in neural information processing systems, 17.

Henderson, P., Islam, R., Bachman, P., Pineau, J., Precup, D. and Meger, D. (2018) Deep reinforcement learning that matters. In: Anonymous Proceedings of the AAAI conference on artificial intelligence.

Howard, R.A. (1960) Dynamic programming and markov processes.

Li, Y. (2017) Deep reinforcement learning: An overview. arXiv preprint arXiv:1701.07274,.

Watkins, C.J. and Dayan, P. (1992) Q-learning. Machine Learning, 8 (3) 279-292.

6 Appendix

1.

	Runs										
number of pits	1	2	3	4	5	6	7	8	9	10	Average
1	36	27	34	14	26	29	20	28	25	32	27.1
2	34	29	32	27	28	25	24	36	28	34	29.7
3	29	31	33	33	20	40	6	35	24	41	29.2
4	26	15	19	25	25	33	14	41	25	24	24.7
5	24	27	15	17	20	25	24	23	20	24	21.9
6	28	24	22	23	26	18	15	20	18	21	21.5
7	15	17	22	18	21	18	22	25	24	19	20.1
8	17	29	18	17	20	18	15	22	21	19	19.6
9	28	17	15	18	19	15	28	21	16	15	19.2
10	14	17	21	16	22	19	16	18	22	20	18.5

Appendix 1 Average Score table for number of pits

2.

	Runs										
number of bonuses	1	2	3	4	5	6	7	8	9	10	Average
2	29	31	33	33	20	40	6	35	24	41	29.2
4	34	48	52	36	37	39	34	40	36	43	39.9
6	48	56	55	61	49	52	54	57	49	51	53.2
8	54	56	47	62	52	49	59	42	54	58	53.3
10	53	61	39	58	54	62	56	49	57	54	54.3

Appendix 2 Average Score table for number Bonuses

3.

	Runs										
meanie spawn rate	1	2	3	4	5	6	7	8	9	10	Average
1	36	21	3	12	16	14	23	15	13	12	16.5
3	16	28	31	27	29	23	19	26	28	27	25.4
5	28	31	35	33	22	39	32	35	24	41	32
8	28	38	31	38	35	36	32	37	35	38	34.8
11	40	24	33	38	41	37	41	31	36	41	36.2

Appendix 3 Average Score table for meanie spawn rate