ARTIFICIAL INTELLIGENCE ASSESSMENT ITEM 2

CONTROLLING THE ACTION OF TALLON IN AN ARENA GAME USING AI METHODS

Suryashankar B Balaganpathi Bhat (25403914)

content

1.Abstract:

this report explains the implementation of artificial intelligence method in the arena game to control the movement of the player where, the Markov decision process is used to control the movement of the robot for various conditions. The Markov process, value iteration and optimum policy concept is explained with implementation of this methods to control a player talon in an arena game.

2.Introduction:

Making decisions is a fundamentally human behaviour with far-reaching consequences. Researchers have sought to improve the quality of judgments by using computer technologies to supplement and expand human capacities, which is probably unsurprising.

Artificial Intelligence (AI) advancements have made this ambition a reality in a variety of applications. Intelligent decision support systems (IDSS) are AI-integrated decision-making support systems that are increasingly being utilised to help decision making in fields including finance, healthcare, marketing, commerce, command and control, and cybersecurity. The current AI technologies utilised in IDSS are reviewed in this study. Various authors have mentioned such systems in the literature.

The phrase "intelligent" refers to systems that, in some way, resemble human cognitive powers. These systems use artificial intelligence (AI) to reason, learn, recall, plan, and analyse data. AI techniques may be used to augment human capabilities by scanning and picking important data from extremely huge and dispersed data sets [1].

3.Decision making process:

Intelligence, design, decision, and execution are the four steps of the process. The decision maker obtains information and develops a knowledge of the situation during the intelligence phase. During the design process, he or she establishes criteria, creates the model, and examines alternatives. During the choosing phase, a choice or decision is made, and during the implementation phase, the decision maker acts on the decision and learns. The procedure follows a broad sequence, including feedback loops in between steps. The Observe, Orient, Decide, Act is a similar four-step decision-making procedure that researchers in defense-related decision assistance favour. The agent in the AI process act in the similar way it chooses the action based on the conditions given and takes best possible action possible. In this arena game the Tallon should choose the best action possible [2].

3.1Markov decision process:

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. MDPs may be used to investigate optimization issues that are solved using dynamic programming.[3]

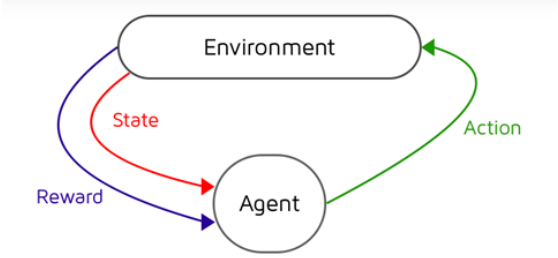

Fig 1: Markov Decision process tree

As shown in the above Fig1 the agent considers the state and reward before taking the next action and choose the optimal action which is provided by optimum policy in our case Tallon is the agent.

s is a 4 tuple (S, A, Pa, Ra) where:

S is the set of states

A is the set of actions

Where P is the probability of movement of the to the of actions

In a Markov decision process, the aim is to develop an appropriate "policy" for the decision maker: a function pi that defines the action (s) pi(s) that the decision maker will take when the states' s is reached. When a Markov decision process and a policy are combined in this way, the action for each state is fixed, and the resulting combination behaves like a Markov chain (since the action chosen in state s s is completely determined by (s) pi(s), and Pr (s t + 1 = s ′ s t = s, a t = a) reduces to Pr (s t + 1 = s ′ s

In this assignment of controlling the movement of the Tallon (player) in the arena game the set of actions are moving north, south, east and west and the set of states are total number of grids which is 100 in our case considering 10x10 grid as shown in Fig 2, Where the black the pits, bonus and meanies are represented.

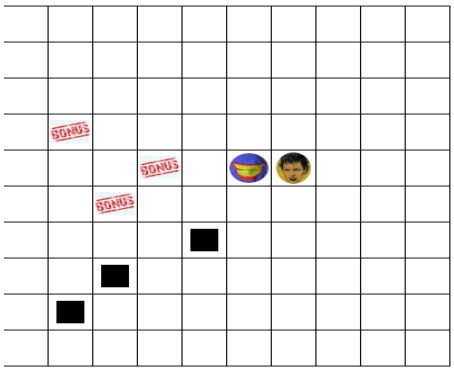

Fig 2: Arena game grid presentation

3.2. Partially observable Markov Decision Process:

A partly observable Markov decision process (POMDP) is a Markov decision process that is broader (MDP). A POMDP is a model of an agent decision process in which the system dynamics are thought to be driven by an MDP, but the agent is unable to perceive the underlying state directly. Instead, it must keep track of the underlying MDP and a sensor model (the probability distribution of distinct observations given the underlying state). POMDP's policy is a mapping from the observations (or belief states) to the actions, unlike MDP's policy function, which maps the underlying states to the actions. A discrete-time POMDP models the relationship between an agent and its environment [3].

A Markovian belief state allows a POMDP to be expressed as a Markov decision process in which each belief represents a state. Even though the "originating" POMDP has a finite number of states, the resultant belief MDP will be defined on a continuous state space: there are infinite belief states (in B) because there are an infinite number of probability distributions over the states (of S). by assuming it as the MDP use the same method to find the optimal policy using the formula same as MDP.

In the arena game where the visibility is partially observable the Tallon try to move randomly and try to find the bonuses using the policy value which is updated every time the Tallon moves.[4]

There are mainly three concepts in MDP to understand it

Utility, policy and value iteration

3.2.1 Utility:

The greatest predicted utility of the state sequences available from that state is added to the reward value for existing in that state.

$U(i) = R(i) + \max(a) \text{ SUM } [M(a,i,j) * U(j)]$

If there is only one possible action a from state i, then

$U(i) = R(i) + \text{SUM } [M(a,i,j) * U(j)]$

3.2.2 Optimal policy: The optimum strategy in a finite Markov Decision Process (MDP) is defined as one that maximises the value of all states at the same time. To put it another way, if there is an optimum policy, the policy that maximises the value of state S is the same as the policy that maximises the value of state S.

This optimum policy will help the Tallon to choose the best action

3.2.3Value iteration: Value iteration is a technique for determining the best MDP policy and its value.

Value iteration begins at the "end" and goes backwards, revising a Q* or V* estimate. It employs an artificial end point because there is no genuine end. Let V_k be the value function and Q_k be the Q-function, assuming there are k stages to go. They can be defined in a recursive manner. To acquire the functions for k+1 stages to go from the functions for k stages to go, value iteration starts with an arbitrary function V0 and employs the following equations:

$Q_{k+1}(s,a) = \sum_{s'} P(s'|s,a) (R(s,a,s')+ \gamma V_k(s'))$
for $k \geq 0$

$V_k(s)= \max_a Q_k(s,a)$ for $k>0$.

We'd need to follow a set of steps. A straight plan, however, would not work since each step is unknown. As a result, we'd need a series of actions for every potential condition. The policy is the name given to this set of acts.

The comprehensive mapping from states to acts is known as policy. The Bellman Update Equation is used to determine the policy in this case.

3.2.4 The Bellman Equation: One of the most important aspects of Markov Decision Processes is the Bellman Equation. It lays out a method for finding the best anticipated reward at a given condition.

According to the value iteration algorithm, the utility $U_t(i)$ of any state i, at any given time step t is given by,
At time t = 0 , $U_t(i) = 0$
At other time , $U_t(i) = \max_a [R(i , a) + \gamma \Sigma_j U_{t-1}(j) P(j|i , a)]$
The above equation is called the Bellman Update equation. Here, we repeat this equation till the model converges. (max change in the utility of than δ, the bellman factor)

3.2.5 Steps carried out while doing value iteration algorithm:

1.Initialise the utilities of all the reachable states as 0.

2.converged=0

3.while(converged==0):

3.1. Calculate the utility of every state for the next iteration, using the Bellman Update Equation

3.2. calculate the maximum difference in utility values between the current iteration and the previous iteration

3.3. If this difference is less than δ, the bellman factor, we say that the model has converged, i.e., converged=1
Once the model has converged, run the algorithm once more, but this time, note down the action that resulted in the maximum utility at each state.

The set of such actions for each state gives us the policy of the given model.
Using the above step, the mdp toolbox will calculate the optimum policy for the Tallon to take the best action.
At first the reward array and the probability array should be made this is done by using the function probability in the code which creates and updates both array every time the function is called and returns both the values of P and R and gives the optimum policy. The directional probability is given
In the code which is 0.95 and the empty cell reward is assumed in the bonus as 1, the rewards assumed in the pits as -1 and the rewards in the empty grids as 0.04.

Since the Partial observable is considered as MDP all the process mentioned above are used to control Tallon's movements under partially observable condition

4. Evaluation of Tallon movements for different condition:

Partially Observable Tallon's score	Fully observable Tallon's score
19	**36**
31	**25**
22	**45**
3	**15**
31	**40**
4	**25**
31	**40**
4	**15**
14	**36**
31	**41**

Table 1: Tallon's score based on visibility

As shown in the **Table 1** when the partial visibility is set to false it becomes fully observable means it can sense all the things around the grid hence the values of the final score mentioned is usually more this means it can go to the bonus location and collect it.
When the partial visibility is set true it can only sense 6 cells from the current location if it finds the bonus it will go and collect it but if the bonus is far away it will try to move randomly to find the bonus. So, it may take a while and sometimes meanies catch the Tallon before Tallon reaching the bonus.
When the meanies visibility is set to 2 then the Tallon's score was more because the meanies senses became less. Similarly, by reducing the number of pits or reducing the incremental value in increasing the meanies will result in the better final score.
When the grid sizes are very small then the Tallon died quickly because of lack of space to move.
The Tallon gets stuck sometimes when the visibility is partially observable, this is due to random location selected in the arena.
When the directional probability is decreased to 0.5, the Tallon moved like it is stuck between two grids. It is because the directional probability gives the probability of movement of the Tallon into the intended direction by

changing it. it has the probability of moving in any other 3 directions.
When the meanies are near the Tallon it tries to escape from the meanies instead of collecting the bonuses.

5.Result:
The control action for the Tallon is provided in tallon.py file using Markov decision process AI method and value iteration algorithm for both partially observable and fully observable process using the single code since each time the Tallon moves it updates its probability and reward array is updated, then the new optimum policy is obtained which gives the Tallon the best possible action.

6.Conclusion:
The AI method is implemented to control the action of Tallon the Markov decision process is used in this assessment since In the Partial observable Markov decision process assumes the process as the Markov decision process the same code is used for controlling the action of the Tallon for both the conditions.

7.References:

[1].Phillips-Wren, G. and Jain, L., 2006, October. Artificial intelligence for decision making. In International Conference on Knowledge-Based and Intelligent Information and Engineering Systems (pp. 531-536). Springer, Berlin, Heidelberg.

[2]. Weiss, G. ed., 1999. Multiagent systems: a modern approach to distributed artificial intelligence. MIT press.

[3].Poole, D. & Mackworth, A. (2017), Artificial Intelligence: Foundations of Computational Agents, Cambridge University Press, Cambridge, UK.

[4]. Puterman, M.L., 2014. Markov decision processes: discrete stochastic dynamic programming. John Wiley & Sons.

[5].Koller, D. and Parr, R., 2013. Policy iteration for factored MDPs. arXiv preprint arXiv:1301.3869.