CS405 Introduction to Empirical Modelling

Coursework topic:

AI with EM, OXO case study

Student ID 0745678

I. Abstract

The ultimate goal of AI research is to make a machine, which can generate a result which is identical to human work or cannot be identified between its result and human's result.

In this document, I will present my new way to approach this ultimate goal with the support of EM. My target is to create a mind map which is base on some pre-defined rule. The dependency of states on this mind map can be auto re-defined by the AI its self, base on its experience during its life time.

I use OXO game model as my case study, in this document I will explain the structure of this model, and how it work.

II. A new approach of AI

II.1 The limitation of current AI study

"Artificial intelligence (AI) is the study and design of intelligent agents, where an intelligent agent is a system that perceives its environment and takes actions which maximize its chances of success" (Wikipedia).

According to this definition, the ability of AI is the ability of making decision which is lead to a best result.

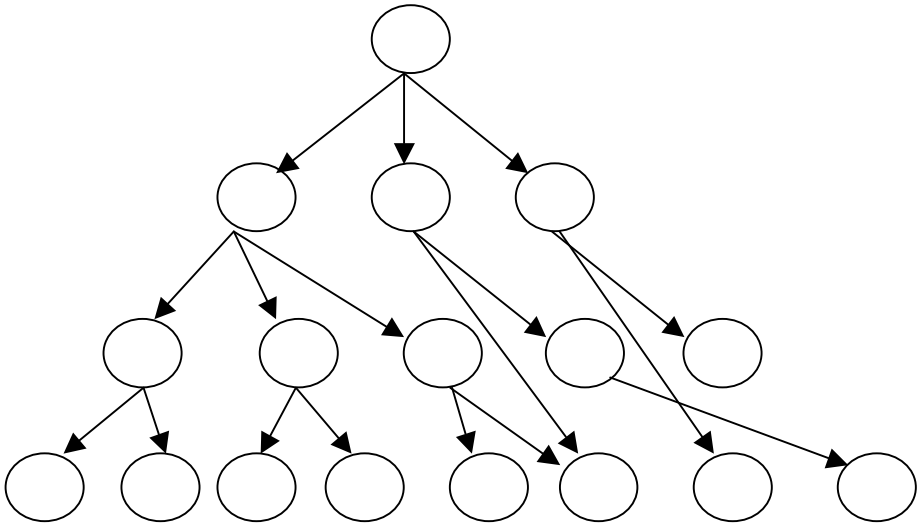

Figure 1.

From figure1 we can image an AI as a decision tree, with lots of IF... THEN... ELSE. We go from the top of the tree, and base on the input, we then result in one of the node at the bottom of the tree. The more accuracy and detailed the tree is designed, the better result we can get.

It means we (the designer) have to think of all the possible state before we create the AI. In another word: the AI cannot think, we have to think for it, and it only apply the input to the suitable track we have draw for it.

II.2 Draw a triangle experiment

When study AI, I try to find out the difference between the way a human think, and a computer, and what make two peoples think difference.

I come up with a simple experiment: we all know what a triangle is, and the definition for it. I then create a program that draw a triangle, and then I myself draw a triangle on a paper, and ask my friend to draw another one. The result is that my triangle and my friend's triangle are not the same, and the triangle my program draw are always the same however I restart this program and re-run it many times.

It is because when I draw a triangle, I only care about the definition of a triangle, and I try to obey the triangle's rule. The position of the vertices and the length of the edges change every time I re-draw the triangle. It depends on many factors that affect me before and during my draw.

I and my friend we know the same rule, but we experienced difference things in our life, so we have difference results.

My program is difference, it does not know the definition of the triangle, what it does is obey the list of command entered to it, and execute it one by one. So as long as those commands are not changed, the result is not changed.

II.3 My idea for AI

Base on my "draw a triangle experiment", I come up with the idea for a new way of AI design. Instead of creating a decision tree like figure1, we define a set of rule, and the AI has to use those rules to define the decision tree itself, and this tree can be changed, re-defined depend on the experience of the program during it runtime.

III. OXO model

To explain my idea more clearly, I will use OXO model as the case study. OXO game is a two players game, each player draw a X or O on one of 9 squares on a 3x3 board. The player get 3 squares on the same line (horizontal, vertical or diagonal) win the game.

To run my model, load run1.e in tkeden.

III.1 The rule set

Like I have mentioned in previous part, instead of define a decision tree, I defined a set of rule which can be used to define a decision tree.
My rule for the oxo model is that:

- The AI know the winning/losing/drawing condition
- The AI want to win, and hate to lose. If It reach the winning state it will be happy.
- If AI reach a state that lead to a winning state, it also happy.
- The longer the path from current state to winning state the less happy the AI is.
- AI only know the state that has been played
- AI cannot deduce a move base on any other rule outside of the rules listed above.

- When It cannot deduce a move base on the rules listed above, it will take the first possible move.

Using those rules, I have played again an existed OXO model on EM website, in the first game I lose, because I don't know which move could lead me to win, and I choose the first possible move. After a few game, I have know which move can make my state better, and choose the best possible move. My "knowledge" about each move is updated every time I make a new move, and it improve my play.

III.2 "board.d" , the Donald model

This file have two viewport, the first view port is board, where I define the table, with 4 horizontal lines and 4 vertical lines.

The second viewport is mindmap. A mind map consists of many difference openshape statex. Each statex can be thought of as a point on the mind map, and it stand for a state has been memorised by the AI. Each statex have the variable to link it to its successor, and two special variables: toWin, toLose tell us the minimum steps from this state to winning state or losing state.

But at the beginning the mindmap is blank.

III.3 "monitor.s" , the SCOUT model

This file defines a monitor for viewport board, and a button to play a new game. It also define 9 text boxes for each square on the board, these text boxes hold the state of the table. For example if we have an X on square1, then text box 1 will have string value = "X".

III.4 "game.e" , EDEN model

This file consist of many procedures that make change to table's state and mind map's state according to the action entered from mouse.

Reset() procedure will clear the table, when there is a click on "new game" button.

humanMove() procedure : when user clicked on a blank square on the table, the procedure will then recorded the current state as "lastState", and draw an "X" on the clicked square. After that It call AIMove() procedure.

AImove() procedure: this procedure will check if the current state has been memorised in mind maps, if it has not been memorised in mind maps then it will create a new plot for this state on the mind map.

After this, AImove() will add this plot as a successor of the plot of "lastState". It also re-define the dependency of lastState toWin and toLose. They will be the minimum toWin and toLose of all of its successor + 1.

AIMove() then check the winning condition of the board, if the condition is met, it then set the toWin or toLose of current state to 0 (toWin =0 mean this is winning state, and toLose = 0 mean this is losing state).

If the winning condition has not been met, AIMove() will generate a move which is lead to a state which is closest to a winning state, and farthest from losing state (base on the mind maps). If mind maps cannot helps to deduce a state like this, then AIMove() will make a first possible move.

After a new move has been made, AImove re-define the mind map again.

III.5 testing result

On the first game, I can win this model easily by placing three X on (1,1), (2,1),(3,1).

I press "new game", and do the same, This time the AI tried not to use the same move like last time, but it still lose.

After few games (about 10 games), now I cannot win by using the same move again. So the AI has "learned".

III.6 Advantage of this model

In this model, the move is generated by using the mind map. But the mind map is blank at the beginning. And it is re-defined each time a new move is made. The mind map at current time is difference from the mind maps of last game, and will be difference from the mind maps of next game. It grows with "experience".

If this model is run on two difference machine, and live two difference experiences, then they will generate two difference mind maps.

The status of each state on the mind maps are updated automatically, thanks to the feature of EM.

III.7 what can be done better?

With this model I have successful with my idea about define an AI using set of rules rather than a decision tree. But the AI is still not really smart, we can improve this AI by adding move rule to it's, for example:

- Two states which are identical when we rotate the table can be treated as 1 state.
- Human can forget, so if a state has not been revisited for a long time can be deleted from the mind maps.

IV. References

All the lecture notes and papers from EM website.