Project Title: Constructive AI with EM

Student ID: 0609155

Abstract

The motivation of this project comes from the idea of using computer to mimic human thoughts and behaviors. That is based on a set of predefined rules and observable environment and experience the computer is able to 'reason' build their own 'knowledge' and make a decision. This project focuses on how the of modeling human knowledge and learning behavior, thus creating a learning machine, can become a very profound approach to Artificial Intelligence. With the help of EM, I use the Noughts and Crosses game as the base study and my approach in studying and derive a strategy for the game, build a model that can, based on its experience, generating its own logic tree and play the game.

Introduction

How does we human observe, perceive the environment, process information, make logical reasoning and memorize them? I believe the process of human learning is the constant exchange of information between our pool of knowledge and the outside observable worlds. The only way for human to interact with the environment is through our senses. Logic is also introduced via senses. At the moment, there is no consensus on how the brain should be stimulated. Computers lack the ability to observe, to make sense of the observable. However, in reasoning over the same set of objects, rules and dependencies the computers have many advantages. These advantages can be used to create a new AI model.

The formal study of AI is to build intelligent agents which can and make decision based on its perceived environment and produce the best outcomes. One of the most common approaches to AI is to create a full decision tree for all the states the machine can be in throughout its life time and the optimal actions to take in it states. Every possibility for the machine states is coded in the machine memory by human developer and thus the machine does exactly as it is told to do. This approach can be use very effectively in a small and perfectly observable environment. For a small game like Noughts and Crosses, the decision tree can be generated very quickly and thoroughly. Any game between a human and a decent AI for the game would likely to result in a draw.

The approach I would like to present here take a step back from the conventional way of developing an AI, instead of telling the computer what to 'do' in every given situations, the computer is told how to 'reason' in each situation. So the computer will complete the developer's work of generating such decision trees based on how the developer wants it to 'learn'. Essentially, this new approach would eventually provide the same result as the conventional way for deterministic, finite, fully observable game like Noughts and Crosses. However, this approach to AI is powerful in cases where human can not anticipate all the

outcomes of the situation. For example, in the game of Go which play in a 36*36 board with very simple rules, however because of the large board the complexity becomes very difficult. Currently the best AI for the game can only achieve lower-intermediate level. If this approach to AI is applied for the game of Go, and after lets the machine 'learn' the game by playing with other human or replaying old games, the result can be proved to be very promising.

The concepts of EM revolve around objects, dependencies and agents. Many successful models have been built based on EM concepts; I believe these concepts are essential and can be used to represent human knowledge. This project makes an attempt of using EM to model human learning process. Based on the previous idea, I used the Noughts and Crosses game as the environment to build a learning machine and achieved very interesting result.

The constructive AI model design

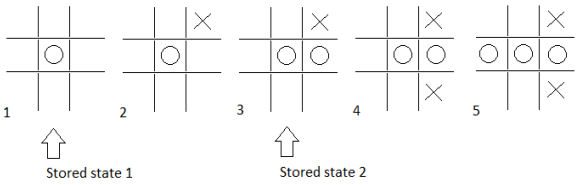
Objective of my model is that the computer can, with a given set of pre-defined rules, objects and their dependencies can create its own decision trees, define and query states of the game, modify the actions in between states based on the outcomes of the game.

Preliminary knowledge

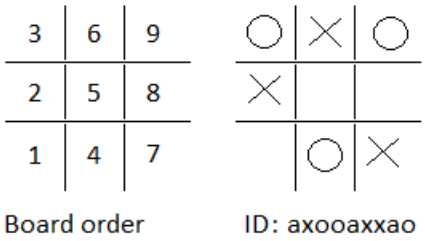
The computer understands the layout of the board, can read in all relevant information of the board, observe all the moves made in game, knows how to place a piece and knows the winning, losing and drawing conditions.

Decision tree creation

The computer only create and store a new state in the decision tree if the state has never been created before and it only create a state when it is the computer turn to make a move. For example, if the game ends in 5 moves made by both players, the human wins using 3 moves, and the computer lost after 2 moves, only 2 states is stored and created by the computer that is before it made its move. Only 2 new states is created because it is necessary to keep track of all the changes in the board using the move list made by both players and only the states before AI make a move is critical in changing the game's outcome.



Each state in the decision tree is identify by the order of the pieces, a indicates available slots, x indicates slots with x pieces and o indicates slots with o pieces, in the order shown below, e.g. axooaxxao shows



The maximum number of states the game can contain is (9*8*7*6*5*4*3*2*1)/2 because only half the states are stored.

The key idea to this approach is the machine capability to create, query and modify states of the system which is very well support by Empirical Modeling tools by using *execute(cmd)* command. By concatenate the state ID with the

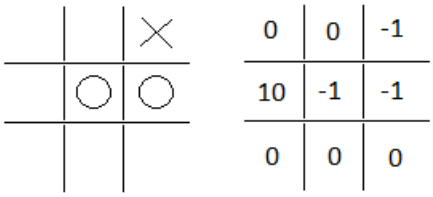
variable names in *cmd* new objects can be created and query directly.

Decision tree modification

Each state stored in the decision tree has a matching values array for each possible moves ranging from 0-10 and -1 for occupied slots. The higher the number the more preferable the outcomes will be. So each time the computer reach a state that it has encounter before in its lifetime, the computer would choose the highest possible value slot to make a move for the best outcome.

Based on Donald Michie's ideas on the machine learning Noughts and Crosses and his model using the matchboxes and the glass beads [3], where the matchboxes are the states stored by the computer and the glass beads is used in the same way as the value table in this case, I implement a different point rewarding system that involve more states than Donald Michie's that can relatively make the computer 'learn' the game faster. Donald Michie's model involves changes to the 300 states that are strictly 4 moves into the game, in this model I implement so far 5 rules that make changes to 2 last states the computer encounter (can be extended easily) thus can cover more than 300 states mention in Michie's model, however, not all states in Michie's model are covered. The first 3 rules are similar to Michie's model which is: reward when the computer wins or draw and set penalty when the computer loses. 2 additional rules I introduced are that if the move put the computer in a bad position i.e. the state could lead to a lost next move, the value for that move is decremented. And to avoid dual winning move by opponent, the computer would try to block the move earlier. The computer can also mimic the opponent moves thus reducing their advantages.

An example of a state and its value indicates blocking opponent's move and minimize mistakes



The image show the board position and what the values table would be after the human win by a move to slot 2, the value of that slot , increase to 10 in the trees, and the value of computer's last (wrong) move is reset to 0.

The Model Implementation

This model I implemented using Donald to define the layout of the board, Scout to control screen output and Eden to create all objects dependencies and the AI learning algorithms

Donald model

The model contains 3 viewports, 1 main viewport define the layout drawing and pieces of the board and 2 viewports to define layout of the 2 states of computer behavior in creating its decision map. The first viewport contain 8 points to draw 4 lines of the board, 9 centre points where the pieces should be drawn and two X and O shapes. Initially, the shapes are drawn out side of the board then are moved to appropriate positions when necessary. The 2 viewports for computer status only contain 8 points to draw the board layout. The number

represent in each square is controlled using Eden.

Scout model

This model controls the information display on screen. Display 3 viewports of the Donald model on screen and defining 18 values to be display on 2 computer status viewports. These 18 values are managed in the Eden model.

Eden model

This model is the main control of the game. Create all the dependencies of the Donald and Scout model with the actual data of the game. The main control function is when the user clicked on the game board, it will perform the win/lose/draw checks as well as make appropriate calls to function generate AI move and human move.

Function *AI()* generate a move for AI consist of 3 main steps. First is to create a new state for the decision map if is not existed *generateStateID() the* query the current state in the decision tree and make a move decision. A random move is made if it is a new states or no best move is found. When the a game is finished, function updateMap() makes necessary update to the existing decision tree using 5 rules define above.

Note on "AI.e". I defined the values for newly defined states not evenly to observe the computer behavior easier. The available square values of all squares in a new state should be 0. However, I set it up based on the number of connections each squares has.

2	0	4
0	5	0
1	0	3

The computer can learn the game faster this way. However, not all possible states are covered. These values can be changed back using vi1 to vi9 define in the beginning of "AI.e"

Result and possible extensions

Model result

Using the optimized value tables, the model play quite decently after about 15 games, it managed to get 4-5 draws with me. Using normal value, it takes longer for the model to learn the game but all states are considered.

Possible Extension

- Currently, I manage to develop two links that is the computer can look back up to two previous steps and adjust the value based on the outcomes. In this model every state can be traced and re-defined however, I have not defined the rule for the computer to do so.
- New function can be introduced to let the AI play against each other and therefore, self learnt the game.
- Using more flexible ID for states the computer can create a full decision tree of the game, and from there it can not only take actions in regarding to its own move, but can also mimic human action if the action has positive outcome

Conclusion

The model is able to achieve interest result based on the learning algorithm. The computer was able to 'study' and derive logic for the game. And the concepts of EM can be used very successfully in modeling the continuous process of human learning.

Reference

[1] OXO model - Garner 1999

[2] AI with EM, OXO case study – EM-04

[3] Donald Michie's MENACE

http://www.adit.co.uk/html/noughts_and_crosses.html