Artificial Intelligence(AI)

CS6659 - AI Notes

Home

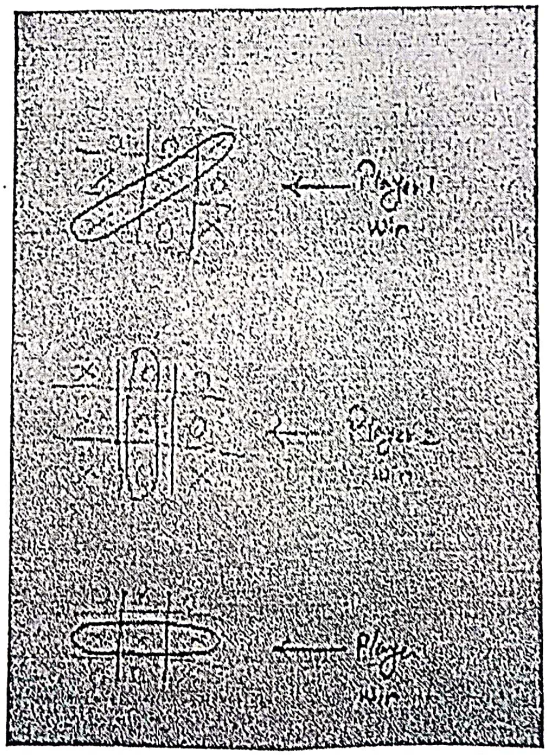
Home

Introductory Problem-Tic Tac Toe

The game **Tic Tac Toe** is also known as **Noughts** and **Crosses** or **X**s and **O**s ,the player needs to take turns marking the spaces in a 3x3 grid with their own marks,if 3 consecutive marks (**Horizontal, Vertical,Diagonal**) are formed then the player who owns these moves get won. We present here three approaches to play this gam which increases in a) complexity b) use of generalization c) clarity of their knowledge d) Extensibility of their approach

Assume ,

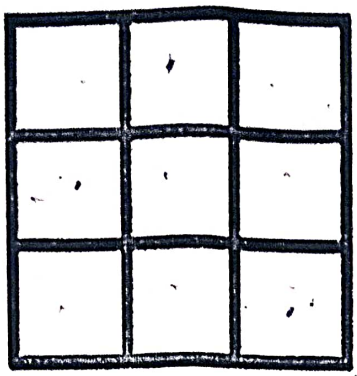
Player 1 - X
Player 2 - O



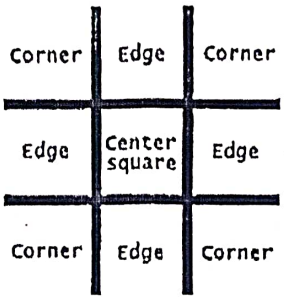
So,a player who gets 3 consecutive marks first,they will win the game .

Let's have a discussion about how a board's data structure looks and how the Tic Tac Toe algorithm works.

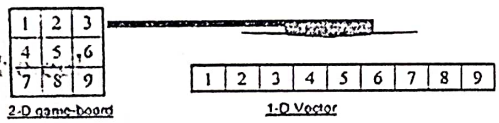
Board's Data Structure:



The cells could be represent as Center square,Corner,Edge as like below



Number each square starts from 1 to 9 like following image



Consider a Board having nine elements vector.Each element will contain

- 0 for blank
- 1 indicating X player move
- 2 indicating O player move

Computer may play as X or O player. First player is always X.

Move Table

It is a vector of 3^9 elements, each element of which is a nine element vector representing board position.

Total of 3^9(19683) elements in move table

Move Table

Index	Current Board position	New Board position
0	000000000	000010000
1	000000001	020000001
2	000000002	000100002
3	000000010	002000010

* The entries of the table are carefully designed manually in advance keeping in mind that Computer should never lose.

* Each entry is indexed by decimal representation of current board position digits.

* Initially board is Empty

* All the possible Board positions are stored in "Current Board position" column along with its corresponding next best possible board position in "New board position column".

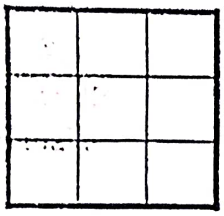
* once the table is designed, the computer programs has to simply do the table lookup. let us write algorithm for the version of a program as follow

Algorithm

To make a move, do the following:

1. View the vector (board) as a ternary number and convert it to its corresponding decimal number.
2. Use the computed number as an index into the move table and access the vector stored there.
3. The vector selected in step 2 represents the way the board will look after the move that should be made. So set board equal to that vector.

Let's start with empty board

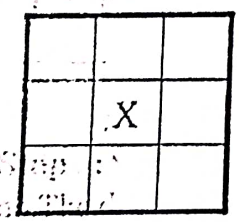


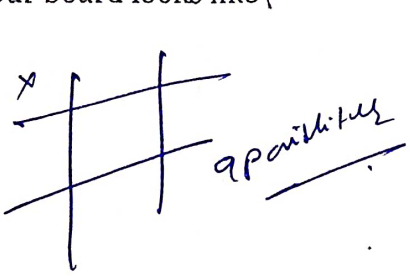
Step 1: Now our board looks like **000 000 000** (tenary number) convert it into decimal no.The decimal no is **0**

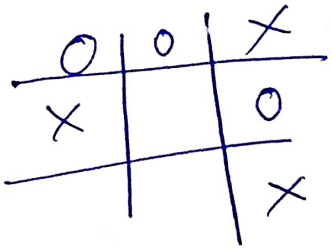
Step 2: Use the computed number ie **0** as an index into the move table and access the vector stored in
New Board Position.
The new board position is **000 010 000**

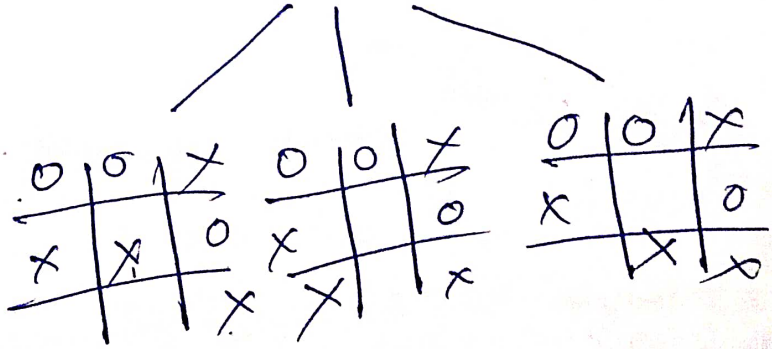
Step 3: The vector selected in step 2(**000 010 000**) represents the way the board will look after the move that should be made. So set board equal to that vector.
After complete the 3rd step your board looks like\

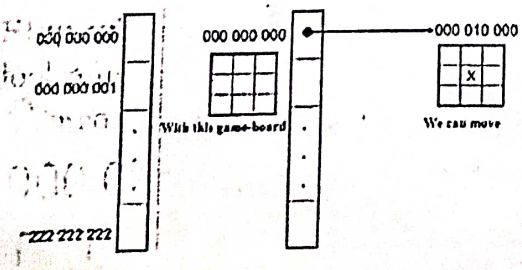
000 010 000











Disadvantages:-

This version of the program is very efficient in terms of time but has several disadvantages.

1. It requires lot of memory space to store the move table
2. To specify entries in move table manually, lot of work is required
3. Creating move table is highly error prone as data to be entered is voluminous
4. This approach cannot be extended to 3D tic-tac-toe. In this 3^{27} board position are to be stored.
5. This program is not intelligent at all as it does not meet any of AI requirements.

Let us develop 2nd version of the program using approach 2 which makes use of human style of playing game. Here again the board is represented by 9 element Vector. we hard code the fact that initially computer at its chance plays in the center, if possible otherwise tries the various non-corner squares.

Approach - 2

The board B[1 --- 9] is represented by a nine-element vector. In this case we choose the following digits to represent blank, X player and O player moves. The choice of these digits will be clear in the strategy part given below.

2 - Representing blank position
3 - indicating X player move
5 - indicating O player move

All 9 moves represented by an integer 1 (first move) to 9 (last move). We will use the following three sub procedures.

* Go(n) → Using this function Computer can make a move in square n
* Make_2 → This function helps the Computer to make valid 2 moves.
* PossWin(P) → If player P can win in the next move then it returns the index from (1 to 9) of the square that constitutes a winning move, otherwise it returns 0.

Strategy:-

The strategy applied by human for this game is that if human is winning in the next ~~wi~~ move then he/she plays in the desired square. else if, human is not winning in the next move then one checks if the opponent is winning. If so then block the square, otherwise try making valid two in any row, column or diagonal.

The function "PossWin" operates by checking, one at a time, for each of rows/columns and diagonals.

✓ If PossWin(P) = 0, then P cannot win. Find whether opponent can win. If so then block it. This can be achieved as follows.

~~If (3*3*2=18)~~

→ If (3×3×2 = 18) then x player can win as there is one blank square in row, column or diagonal

→ If (5×5×2 = 50) then O player can win.

Let us represent computer by C and human by H. The player who is playing first will called X player. Since computer can be first or second player, Table 1.2 consists of rules to be applied by computer for all nine moves. If C is the first player (playing X), then odd moves are to be choosen otherwise, if C is playing as second player (playing O) then even moves are the ones to be followed by C.

Comments in the Rules are enclosed ~~by~~ in { }

Rules for Nine Moves (8)

(C plays X, H plays O)

move: Go(5) / Go(1)

move: If B[9] is blank
then
Go(9)
else
{make 2}
Go(3)

move {By now both have played 2 moves}:
If PossWin(X)
then
{X wins}
Go(PossWin(X))
else if
PossWin(O)
{block(O)}
then
Go(PossWin(O))
else if B[7] is blank
then
Go(7)
else
Go(3)

(H Plays X, C plays O)

2 move: If B[5] is blank,
then
Go(5)
else
Go(1)

4 move: {By now human (playing X has played 2 moves}:
If PossWin(X)
then
{block X}
Go(PossWin(X))
else
{make 2}
Go(Make_2)

6: move: {By now computer has played 2 moves}:
If PossWin(O)
then
{O wins}
Go(PossWin(O))
else if
PossWin(X)
{block(X)}
then
Go(PossWin(X))
else
Go(Make_2)

1 & 9 moves:

{By now human (playing O) has played 3 chances}

If PossWin (X)
then
{X wins}
Go (PossWin(X))
else if
{block O}

If PossWin (O)
then
Go (PossWin(O))
else
Go (Anywhere)

8 move:- (a)

{By now computer has played 3 chances}:

If PossWin (O)
then
{O wins}
Go (PossWin (O))
else
{block X}

If PossWin (X)
then
Go (PossWin (X))
else
Go (Anywhere)

This version of the program is memory efficient and easier to understand as complete strategy has been determined in advance but: has several disadvantages as listed below. It applies heuristics (thumb rules), so can be treated as one step towards AI approach.

Dis-adv:-

1) Not as efficient as 1st one with respect to time several conditions are checked before each move
2) still cannot generalize to 3-D.

UNIT –I Introduction to Artificial Intelligence:

Introduction, History, Intelligent Systems, Foundations of AI, Applications, Tic-tac toe game playing, development of AI languages, Current trends in AI.

1.Artificial Intelligence

- Modeling human cognition or mental faculties using computers
- Study of making computers do things which at the moment people are better
- Making computers do things which require intelligence

1.1More Formal Definition of AI

- AI is a branch of computer science which is concerned with the study and creation of computer systems that exhibit
  - some form of intelligence

  OR
  - those characteristics which we associate with intelligence in human behavior
- AI is a broad area consisting of different fields, from machine vision, expert systems to the creation of machines that can "think".
- In order to classify machines as "thinking", it is necessary to define intelligence.

2.History

The history of Artificial Intelligence (AI) began in antiquity, with myths, stories and rumors of artificial beings endowed with intelligence or consciousness by master craftsmen. The seeds of modern AI were planted by classical philosophers who attempted to describe the process of human thinking as the mechanical manipulation of symbols. This work culminated in the invention of the programmable digital computer in the 1940s, a machine based on the abstract essence of mathematical reasoning. This device and the ideas behind it inspired a handful of scientists to begin seriously discussing the possibility of building an electronic brain.

The field of AI research was founded at a workshop held on the campus of Dartmouth College during the summer of 1956. Those who attended would become the leaders of AI research for decades. Many of them predicted that a

machine as intelligent as a human being would exist in no more than a generation and they were given millions of dollars to make this vision come true.

Eventually, it became obvious that they had grossly underestimated the difficulty of the project. In 1973, in response to the criticism from James Lighthill and ongoing pressure from congress, the U.S. and British Governments stopped funding undirected research into artificial intelligence, and the difficult years that followed would later be known as an "AI winter". Seven years later, a visionary initiative by the Japanese Government inspired governments and industry to provide AI with billions of dollars, but by the late 80s the investors became disillusioned by the absence of the needed computer power (hardware) and withdrew funding again.

In the 1980s a form of AI program called "expert systems" was adopted by corporations around the world and knowledge became the focus of mainstream AI research. In those same years, the Japanese government aggressively funded AI with its fifth generation computer project. Another encouraging event in the early 1980s was the revival of connectionism in the work of John Hopfield and David Rumelhart. Once again, AI had achieved success.

In the first decades of the 21st century, access to large amounts of data (known as "big data"), cheaper and faster computers and advanced machine learning techniques were successfully applied to many problems throughout the economy. In fact, McKinsey Global Institute estimated in their famous paper "Big data: The next frontier for innovation, competition, and productivity" that "by 2009, nearly all sectors in the US economy had at least an average of 200 terabytes of stored data".

By 2016, the market for AI-related products, hardware, and software reached more than 8 billion dollars, and the New York Times reported that interest in AI had reached a "frenzy". The applications of big data began to reach into other fields as well, such as training models in ecology and for various applications in economics. Advances in deep learning (particularly deep convolutional neural networks and recurrent neural networks) drove progress and research in image and video processing, text analysis, and even speech recognition.

Investment and interest in AI boomed in the first decades of the 21st century, when machine learning was successfully applied to many problems in academia and industry due to new methods, the application of powerful computer hardware, and the collection of immense data sets.

3. What is Intelligence?

- Intelligence is a property of mind that encompasses many related mental abilities, such as the capabilities to

- reason
- plan
- solve problems
- think abstractly
- comprehend ideas and language and
- learn

Examples of AI Problem & Application Areas

(1) Expert consulting system
(2) Theorem proving
(3) Robotics
(4) Automatic programming
(5) Perceptional problem
(6) Natural language processing

3.1. Characteristics of AI systems

- learn new concepts and tasks
- reason and draw useful conclusions about the world around us
  - remember complicated interrelated facts and draw conclusions from them (inference)
- understand a natural language or perceive and comprehend a visual scene
  - look through cameras and see what's there (vision), to move themselves and objects around in the real world (robotics)
- plan sequences of actions to complete a goal
- offer advice based on rules and situations
- may not necessarily imitate human senses and thought processes
  - but indeed, in performing some tasks differently, they may actually exceed human abilities
- capable of performing intelligent tasks effectively and efficiently
- perform tasks that require high levels of intelligence

3.2. Understanding of AI

- AI techniques and ideas seem to be harder to understand than most things in computer science
- AI shows best on complex problems for which general principles don't help much, though there are a few useful general principles

- Artificial intelligence is also difficult to understand by its content.
- Boundaries of AI are not well defined.
- Often it means the advanced software engineering, sophisticated software techniques for hard problems that can't be solved in any easy way.
- AI programs - like people - are usually not perfect, and even make mistakes.
- It often means, nonnumeric ways of solving problems, since people can't handle numbers well.
- Nonnumeric ways are generally "common sense" ways, not necessarily the best ones.
- Understanding of AI also requires an understanding of related terms such as intelligence, knowledge, reasoning, thought, cognition, learning, and a number of other computer related terms.

3.3. Categories of AI System

- Systems that think like humans
- Systems that act like humans
- Systems that think rationally
- Systems that act rationally

3.3.1.Systems that think like humans

- Most of the time it is a black box where we are not clear about our thought process.
- One has to know functioning of brain and its mechanism for prossessing information.
- It is an area of cognitive science.
  - The stimuli are converted into mental representation.
  - Cognitive processes manipulate representation to build new representations that are used to generate actions.
- Neural network is a computing model for processing information similar to brain.

3.3.2.Systems that act like humans

- The overall behaviour of the system should be human like.
- It could be achieved by observation

3.3.3.Systems that think rationally

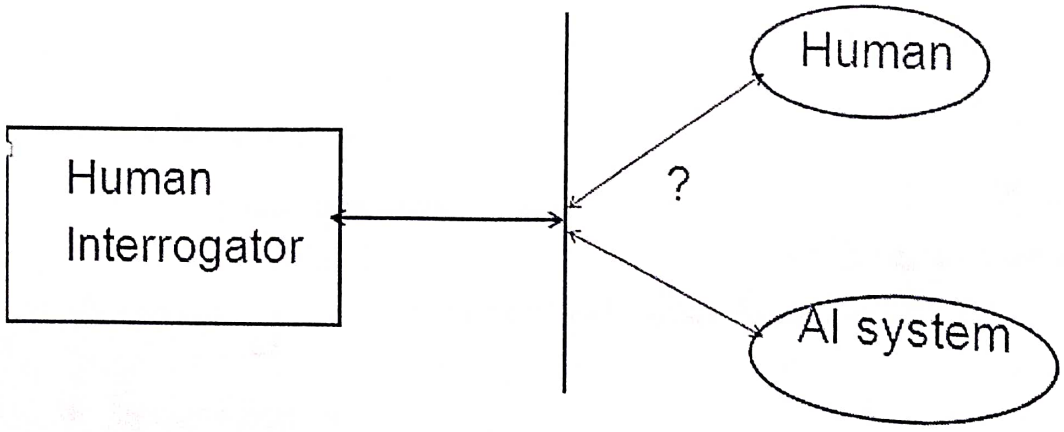
- Such systems rely on logic rather than human to measure correctness.
- For thinking rationally or logically, logic formulas and theories are used for synthesizing outcomes.
- For example,
  - given John is a human and all humans are mortal then one can conclude logically that John is mortal
- Not all intelligent behavior are mediated by logical deliberation.

3.3.4.Systems that act rationally

- Rational behavior means doing right thing.
- Even if method is illogical, the observed behavior must be rational.

3.4.The Turing Test

Turing proposed operational test for intelligent behavior in 1950.



3.5.ELIZA

- Eliza was first program developed by Joseph Wiezbaum to converse in English in mid 1960's
  - It passed turing test.
- The following passage shows Eliza's talking to a teenage girl. Blue text Eliza's response

Men are all alike.

IN WHAT WAY

They're always bugging us about something or other.

CAN YOU THINK OF A SPECIFIC EXAMPLE

Well, my boyfriend made me come here.

YOUR BOYFRIEND MADE YOU COME HERE

He says that I am depressed all the time.

I AM SORRY TO HEAR THAT YOU ARE DEPRESSED

It's true that I am unhappy

DO YOU THINK COMING HERE WILL HELP YOU NOT TO BE UNHAPPY

I need help, that seems certain.

WHAT WOULD IT MEAN IF YOU GOT SOME HELP

Perhaps I could learn to get along with my mother.

TELL ME MORE ABOUT YOUR FAMILY

My mother takes care of me.

WHO ELSE IN YOUR FAMILY TAKES CARE OF YOU

My father.

YOUR FATHER

You are like my father in some ways.

WHAT RESEMBLANCE DO YOU SEE

You are not very aggressive but I think you don't want me to notice that.

WHAT MAKES YOU THINK THAT I AM NOT VERY AGGRESSIVE

You don't argue with me.

WHY DO YOU THINK I DON'T ARGUE WITH YOU

You are afraid of me.

DOES IT PLEASE YOU TO BELIEVE, I AM AFRAID OF YOU

My father is afraid of everybody.

WHAT ELSE COMES TO MIND WHEN YOU THINK OF YOUR FATHER

Bullies.

DOES THAT HAVE ANYTHING TO DO WITH THE FACT THAT YOUR BOYFRIEND MADE YOU COME HERE

4.Foundations of AI

- Foundation of AI is based on
  - Mathematics
  - Neuroscience
  - Control Theory
  - Linguistics

4.1.Foundations – Mathematics

- More formal logical methods
  - Boolean logic
  - Fuzzy logic
- Uncertainty
  - The basis for most modern approaches to handle uncertainty in AI applications can be handled by

- ✓ Probability theory
- ✓ Modal and Temporal logics

4.2.Foundations – Neuroscience

- How do the brain works?
  - Early studies (1824) relied on injured and abnormal people to understand what parts of brain work
  - More recent studies use accurate sensors to correlate brain activity to human thought
    - By monitoring individual neurons, monkeys can now control a computer mouse using thought alone
  - Moore's law states that computers will have as many gates as humans have neurons in 2020
  - How close are we to have a mechanical brain?
    - Parallel computation, remapping, interconnections,....

4.3.Foundations – Control Theory

- Machines can modify their behavior in response to the environment (sense/action loop)
  - Water-flow regulator, steam engine governor, thermostat
- The theory of stable feedback systems (1894)
  - Build systems that transition from initial state to goal state with minimum energy
  - In 1950, control theory could only describe linear systems and AI largely rose as a response to this shortcoming

4.4.Foundations – Linguistics

- Speech demonstrates so much of human intelligence

phonology
phonetics
syntax
Semantics
morphology.

- Analysis of human language reveals thought taking place in ways not understood in other settings
  - Children can create sentences they have never heard before
  - Language and thought are believed to be tightly intertwined

4.5.Two Views of AI Goals

- AI is about duplicating what the (human) brain DOES
  - Cognitive Science
- AI is about duplicating what the (human) brain SHOULD do
  - Rationality (doing things logically)

4.6.Cool Stuff in AI

- Game playing agents
- Machine learning
- Speech
- Language
- Vision
- Data Mining
- Web agents

4.7.Useful Stuff

- Medical Diagnosis
- Fraud Detection
- Object Identification
- Space Shuttle Scheduling
- Information Retrieval

5.AI Techniques

- Rule-based
- Fuzzy Logic
- Neural Networks
- Genetic Algorithms

6.Components of AI Program

- AI techniques must be independent of the problem domain as far as possible.
- AI program should have
  - knowledge base
  - navigational capability
  - inferencing

6.1.Knowledge Base

- AI programs should be learning in nature and update its knowledge accordingly.
- Knowledge base consists of facts and rules.
- Characteristics of Knowledge:
  - It is voluminous in nature and requires proper structuring
  - It may be incomplete and imprecise
  - It may keep on changing (dynamic)

6.2.Navigational Capability

- Navigational capability contains various control strategies
- Control Strategy
  - determines the rule to be applied
  - some heuristics (thump rule) may be applied

6.3.Inferencing

- Inferencing requires
  - search through knowledge base

    and
  - derive new knowledge

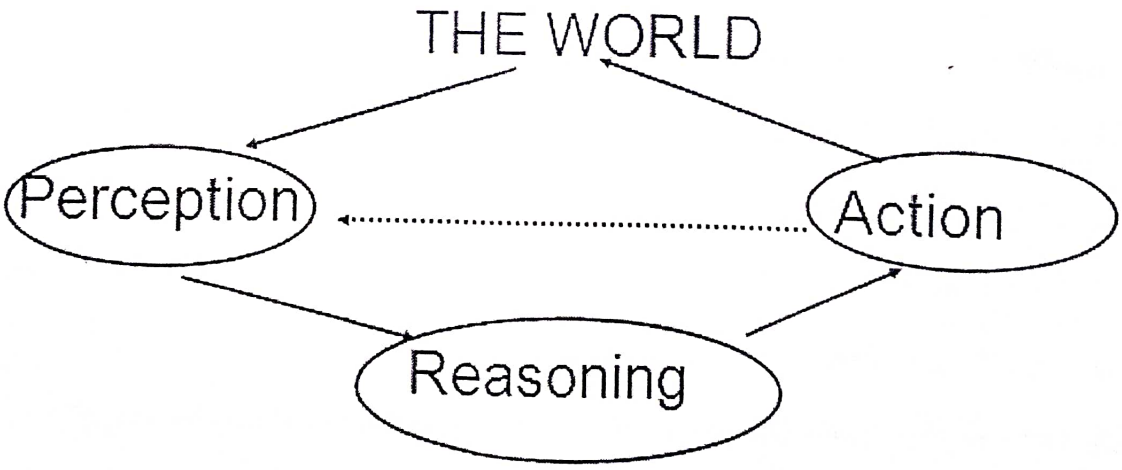
7.Sub-areas of AI

- Sub areas of AI are:
  - Knowledge representation
  - Theorem proving
  - Game playing
  - Vommon sense reasoning dealing with uncertainty and decision making
  - Learning models, inference techniques, pattern recognition, search and matching etc.
  - Logic (fuzzy, temporal, modal) in AI
  - Planning and scheduling
  - Natural language understanding
  - Computer vision
  - Understanding spoken utterances
  - Intelligent tutoring systems
  - Robotics
  - Machine translation systems
  - Expert problem solving
  - Neural Networks, AI tools etc

8.Applications of AI:

- Some of the applications are given below:
  - **Business** : Financial strategies, give advice
  - **Engineering:** check design, offer suggestions to create new product
  - **Manufacturing:** Assembly, inspection & maintenance
  - **Mining:** used when conditions are dangerous
  - **Hospital :** monitoring, diagnosing & prescribing
  - **Education :** In teaching
  - **household :** Advice on cooking, shopping etc.
  - **farming :** prune trees & selectively harvest mixed crops.

8.1.Latest Perception of AI



8.2.Recent AI

- Heavy use of
  - probability theory
  - decision theory
  - statistics

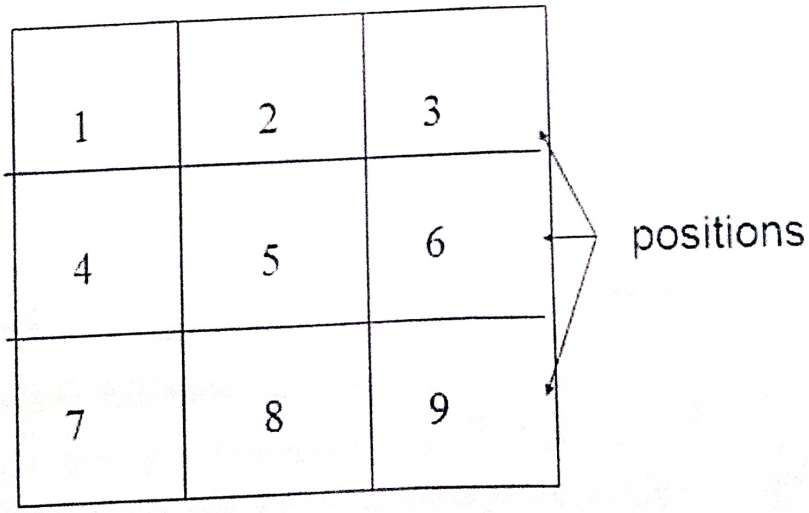
- logic (fuzzy, modal, temporal)

9. Tic Tac Toe Game playing strategies

- Two players
  - human
  - computer.
- The objective is to write a computer program in such a way that computer wins most of the time.
- Three approaches are presented to play this game which increase in
  - Complexity
  - Use of generalization
  - Clarity of their knowledge
  - Extensibility of their approach
- These approaches will move towards being representations of what we will call AI techniques.

Tic Tac Toe Board- (or **Noughts and crosses**, **Xs and Os**)

It is two players, *X* and *O*, game who take turns marking the spaces in a 3×3 grid. The player who succeeds in placing three respective marks in a horizontal, vertical, or diagonal row wins the game.



Approach 1

- Data Structure
  - Consider a Board having nine elements vector.
  - Each element will contain
    - 0 for blank
    - 1 indicating X player move
    - 2 indicating O player move
  - Computer may play as X or O player.
  - First player who so ever is always plays X.

Move Table MT

- MT is a vector of 3^9 elements, each element of which is a nine element vector representing board position.
- Total of 3^9 (19683) elements in MT

Index	Current Board position	New Board position
0	000000000	000010000
1	000000001	020000001
2	000000002	000100002
3	000000010	002000010
⋮		

Algorithm

- To make a move, do the following:
  - View the vector (board) as a ternary number and convert it to its corresponding decimal number.
  - Use the computed number as an index into the MT and access the vector stored there.

- The selected vector represents the way the board will look after the move.

- Set board equal to that vector.

Comments

- Very efficient in terms of time but has several disadvantages.
  - Lot of space to store the move table.
  - Lot of work to specify all the entries in move table.
  - Highly error prone as the data is voluminous.
  - Poor extensibility
    - 3D tic-tac-toe = 3^{27} board position to be stored.
  - Not intelligent at all.

Approach 2

- **Data Structure**
  - **Board**: A nine-element vector representing the board: B[1..9]
  - Following conventions are used

    2 - indicates blank

    3 - X

    5 - 0
  - **Turn**: An integer

    1 - First move

    9 - Last move

Procedures Used

- ***Make_2*** → Tries to make valid 2
  - Make_2 first tries to play in the center if free and returns 5 (square number).

- If not possible, then it tries the various suitable non corner square and returns square number.
- ***Go(n)*** ← makes a move in square 'n' which is blank represented by 2.

Procedure – PossWin

- ***PossWin (P)*** → Returns
  - 0, if player P cannot win in its next move,
  - otherwise the number of square that constitutes a winning move for P.
  - Rule
  - If PossWin (P) = 0 {P can not win} then find whether opponent can win. If so, then block it.

Strategy used by PosWin

- ***PosWin*** checks one at a time, for each rows /columns and diagonals as follows.
  - If 3 * 3 * 2 = 18 then player X can win
  - else if 5 * 5 * 2 = 50 then player O can win
  - These procedures are used in the algorithm are showed below.

Algorithm

- Assumptions
  - The first player always uses symbol X.
  - There are in all 8 moves in the worst case.
  - Computer is represented by C and Human is represented by H.
  - Convention used in algorithm on next slide
    - If C plays first (Computer plays X, Human plays O) - **Odd moves**

- If H plays first (Human plays X, Computer plays O) - **Even moves**
- For the sake of clarity, we use C and H.

Algo - Computer plays first – C plays odd moves

- **Move 1:** Go (5)
- **Move** 2: *H plays*
- **Move 3:** If B[9] is blank, then Go(9) else Go(3) ***{make 2}***
- **Move 4:** *H plays*
- **Move 5:** ***{By now computer has played 2 chances}***
  - If PossWin(C) then ***{won}*** Go(PossWin(C))
  - else ***{block H}*** if PossWin(H) then Go(PossWin(H)) else if B[7] is blank then Go(7) else Go(3)
- **Move 6***: H plays*
- **Moves 7 & 9 :**
  - If PossWin(C) then ***{won}*** Go(PossWin(C))
  - else ***{block H}*** if PossWin(H) then Go(PossWin(H)) else Go(Anywhere)
- **Move 8:** *H plays*

Algo - Human plays first – C plays even moves

- **Move 1:** *H plays*
- **Move** 2: If B[5] is blank, then Go(5) else Go(1)
- **Move 3:** *H plays*
- **Move 4**: ***{By now H has played 2 chances}***
  - If PossWin(H) then ***{block H}*** Go (PossWin(H))
  - else Go (Make_2)
- **Move 5:** *H plays*

- **Move** 6: ***{By now both have played 2 chances}***
  - If PossWin(C) then ***{won}*** Go(PossWin(C))
  - else ***{block H}*** if PossWin(H) then Go(PossWin(H)) else Go(Make_2)
- **Moves 7 & 9 :** *H plays*
- **Move** 8: ***{By now computer has played 3 chances}***
  - If PossWin(C) then ***{won}*** Go(PossWin(C))
  - else ***{block H}*** if PossWin(H) then Go(PossWin(H)) else Go(Anywhere)

Complete Algorithm – Odd moves or even moves for C playing first or second

- **Move 1:** go (5)
- **Move** 2: If B[5] is blank, then Go(5) else Go(1)
- **Move 3:** If B[9] is blank, then Go(9) else Go(3) ***{make 2}***
- **Move 4**: ***{By now human (playing X) has played 2 chances}*** If PossWin(X) then ***{block H}*** Go (PossWin(X)) else Go (Make_2)
- **Move 5:** ***{By now computer has played 2 chances}*** If PossWin(X) then ***{won}*** Go(PossWin(X)) else ***{block H}*** if PossWin(O) then Go(PossWin(O)) else if B[7] is blank then Go(7) else Go(3)
- **Move** 6: ***{By now both have played 2 chances}*** If PossWin(O) then ***{won}*** Go(PossWin(O)) else ***{block H}*** if PossWin(X) then Go(PossWin(X)) else Go(Make_2)
- **Moves 7 & 9 :** ***{By now human (playing O) has played 3 chances}*** If PossWin(X) then ***{won}*** Go(PossWin(X)) else ***{block H}*** if PossWin(O) then Go(PossWin(O)) else Go(Anywhere)
- **Move** 8: ***{By now computer has played 3 chances}*** If PossWin(O) then ***{won}*** Go(PossWin(O)) else ***{block H}*** if PossWin(X) then Go(PossWin(X)) else Go(Anywhere)

Comments

- Not as efficient as first one in terms of time.
- Several conditions are checked before each move.

- It is memory efficient.
- Easier to understand & complete strategy has been determined in advance
- Still can not generalize to 3-D.

Approach 3

- Same as approach 2 except for one change in the representation of the board.
  - Board is considered to be a magic square of size 3 X 3 with 9 blocks numbered by numbers indicated by magic square.
- This representation makes process of checking for a possible win more simple.

Board Layout – Magic Square

- Board Layout as magic square. Each row, column and diagonals add to 15.

Magic Square

8	3	4
1	5	9
6	7	2

Strategy for possible win for one player

- Maintain the list of each player's blocks in which he has played.
  - Consider each pair of blocks that player owns.

- Compute difference D between 15 and the sum of the two blocks.
- If D < 0 or D > 9 then
  - these two blocks are not collinear and so can be ignored
  - otherwise if the block representing difference is blank (i.e., not in either list) then a move in that block will produce a win.

Working Example of algorithm

- Assume that the following lists are maintained up to 3rd move.
- Consider the magic block shown in slide 18.
  - First Player X (Human)

    8 3

  - Second Player O (Computer)

    5

- Strategy is same as in approach 2
  - First check if computer can win.
    - If not then check if opponent can win.
    - If so, then block it and proceed further.
- Steps involved in the play are:
  - First chance, H plays in block numbered as 8
  - Next C plays in block numbered as 5
  - H plays in block numbered 3

Now there is a turn of computer

- Strategy by computer: Since H has played two turns and C has played only one turn, C checks if H can win or not.

- Compute sum of blocks played by H
  - S = 8 + 3 = 11
  - Compute D = 15 – 11 = 4
  - Block 4 is a winning block for H.
  - So block this block and play in block numbered 4.
  - The list of C gets updated with block number 4 as follows:

H 8 3 C 5 4

- Assume that H plays in block numbered 6.
- Now it's a turn of C.
  - C checks, if C can win as follows:
    - Compute sum of blocks played by C
    - S = 5 + 4 = 9
    - Compute D = 15 – 9 = 6
    - Block 6 is not free, so C can not win at this turn.
  - Now check if H can win.
    - Compute sum of new pairs (8, 6) and (3, 6) from the list of H
    - S = 8 + 6 = 14
    - Compute D = 15 – 14 = 1
    - Block 1 is not used by either player, so C plays in block numbered as 1
- The updated lists at 6th move looks as follows:
  - First Player H

8 3 6

  - Second Player C

5 4 1

- Assume that now H plays in 2.
- Using same strategy, C checks its pair (5, 1) and (4, 1) and finds bock numbered as 9 {15-6 = 9}.
- Block 9 is free, so C plays in 9 and win the game.

Comments

- This program will require more time than two others as
  - it has to search a tree representing all possible move sequences before making each move.
- This approach is extensible to handle
  - 3-dimensional tic-tac-toe.
  - games more complicated than tic-tac-toe.

3D Tic Tac Toe (Magic cube)

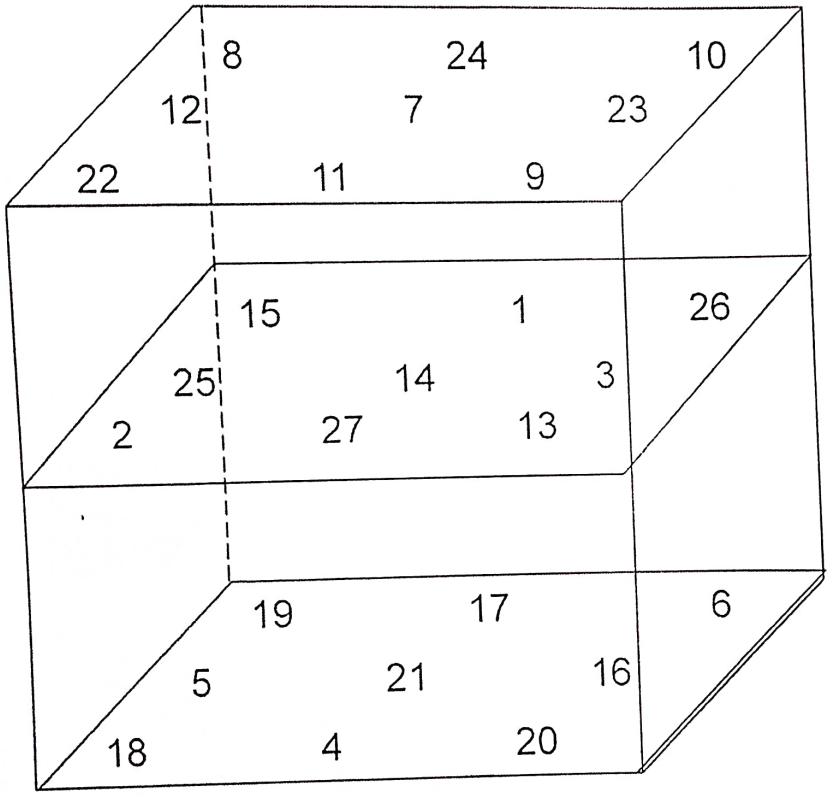
- All lines parallel to the faces of a cube, and all 4 triagonals sum correctly to 42 defined by

$$S = m(m^3 + 1)/2 \text{ , where } m=3$$

- No planar diagonals of outer surfaces sum to 42. so there are probably no magic squares in the cube.

8	24	10		15	1	26		19	17	6
12	7	23		25	14	3		5	21	16
22	11	9		2	27	13		18	4	20

8	24	10		15	1	26		19	17	6
12	7	23		25	14	3		5	21	16
22	11	9		2	27	13		18	4	20



- Magic Cube has 6 outer and 3 inner and 2 diagonal surfaces
- Outer 6 surfaces are not magic squares as diagonals are not added to 42.
- Inner 5 surfaces are magic square.