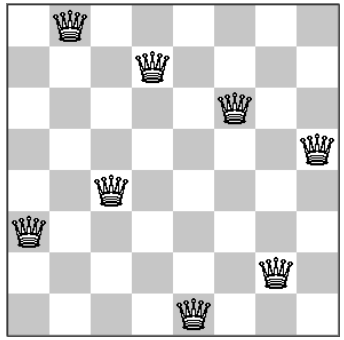
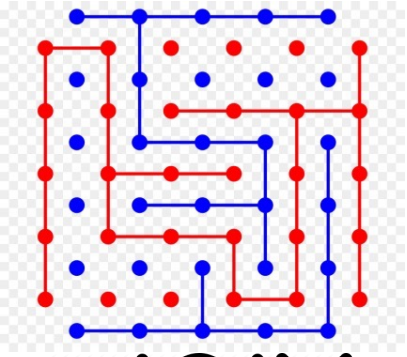
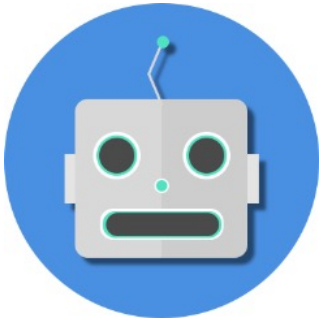
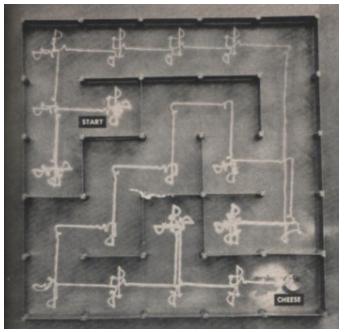
Artificial Intelligence:

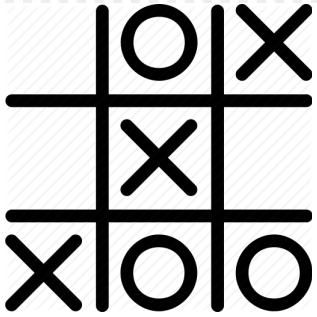
Past, Present and Future











Chee Wei Tan

tive feedback system for hunting for particular solutions of the logical equations without an exhaustive search through all possible combinations. This is achieved by elements which sense whether or not a particular logical relation is satisfied. If not, the truth variables involved in this relation are caused to oscillate between their two possible values. Thus, variables appearing in unsatisfied relations are continually changing, while those appearing only in satisfied relations do not change. If ever all relations are simultaneously satisfied the machine stops at that particular solution. Changing only the variables in unsatisfied relations tends, in a general way, to lead to a solution more rapidly than methodical exhaustion of all cases, but, as is usually the case when feedback is introduced, leads to the possibility of continual oscillation. McCallum and Smith point out the desirability of making the changes of the variables due to the feedback unbalance as random as possible, to enable the machine to escape from periodic paths through various states of the relays.

Game Playing Machines

The problem of designing game-playing machines is fascinating and has received a good deal of attention. The rules of a game provide a sharply limited environment in which a machine may operate, with a clearly defined goal for its activities. The discrete nature of most games matches well the digital computing techniques available without the cumbersome analog-digital conversion necessary in translating our physical environment in the case of manipulating and sensing machines.

Game playing machines may be roughly classified into types in order of increasing sophistication:

1. Dictionary-type machines. Here the proper move of the machine is decided in advance for each possible situation that may arise in the game and listed in a "dictionary" or function table. When a particular position arises, the machine merely looks up the move in the dictionary. Because of the extravagant memory requirements, this rather uninteresting method is only feasible for exceptionally simple games, e.g., tic-tac-toe.
2. Machines using rigorously correct playing formulas. In some games, such as Nim, a complete mathematical theory is known, whereby it is possible to compute by a relatively simple formula, in any position that can be won, a suitable winning move. A mechanization of this formula provides a perfect game player for such games.
3. Machines applying general principles of approximate validity. In most games of interest to humans, no simple exact solution is known, but there are various general principles of play which hold in the majority of positions. This is true of such games as checkers, chess, bridge, poker and the like. Machines may be designed applying such the principles are not infallible, neither are the machines, as indeed, neither are humans.
4. Learning machines. Here the machine is given only the rules of the game and perhaps an elementary strategy of play, together with some method of improving this strategy through experience. Among the many methods that have been suggested for incorporation of learning we have:
   a) trial-and-error with retention of successful and elimination of unsuccessful possibilities;
   b) imitation of a more successful opponent;
   c) "teaching" by approval or disapproval, or by informing the machine of the nature of its mistakes; and finally
   d) self-analysis by the machine of its mistakes in an attempt to devise general principles.

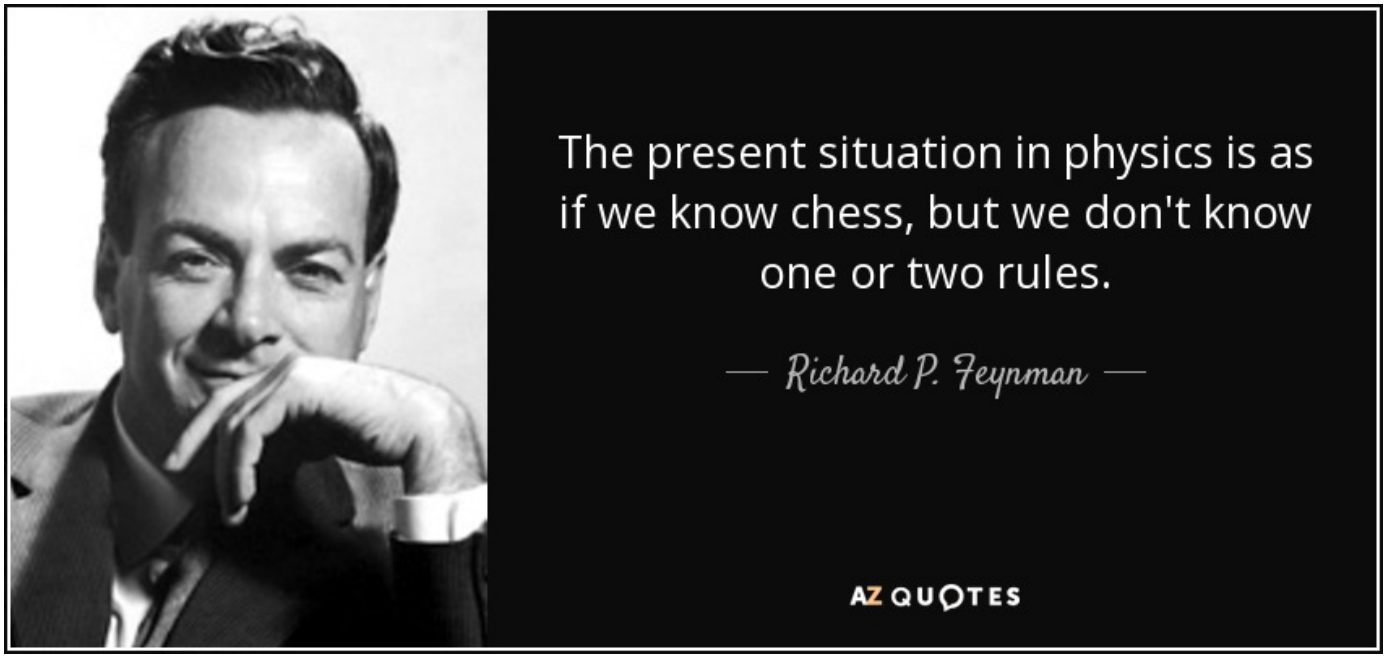
Many examples of the first two types have been constructed and a few of the third. The fourth type, learning game-players, is reminiscent of Mark Twain's comment on the weather. Here is a real challenge for the programmer and machine designer.

Two examples of the third category: machines applying general principles, may be of interest. The first of these is a machine designed by E. F. Moore and the writer for playing a commercial board game known as Hex. This game is played on a board laid out in a regular hexagon pattern, the two players alternately placing black and white pieces in unoccupied hexagons. The entire board forms a rhombus and Black's goal is to connect the top and bottom of this rhombus with a continuous chain of black pieces. White's goal is to connect the two sides of the rhombus with a chain of white pieces. After a study of this game, it was conjectured that a reasonably good move could be made by the following process. A two-dimensional potential field is set up corresponding to the playing board, with white pieces as positive charges and black pieces as negative charges. The top and bottom of the board are negative and the two sides positive. The move to be made corresponds to a certain specified saddle point in this field.

To test this strategy, an analog device was constructed, consisting of a resistance network and gadgetry to locate the saddle points. The general principle, with some improvements suggested by experience, proved to be reasonably sound. With first move, the machine won about seventy per cent of its games against human opponents. It frequently surprised its designers by choosing odd-looking moves which, on analysis, proved sound. We normally think of computers as expert at long involved calculations and poor in generalized value judgments. Paradoxically, the positional judgment of this machine was good; its chief weakness was in end-game combinatorial play. It is also curious that the Hex-player reversed the usual computing procedure in that it solved a basically digital problem by an anlog machine.

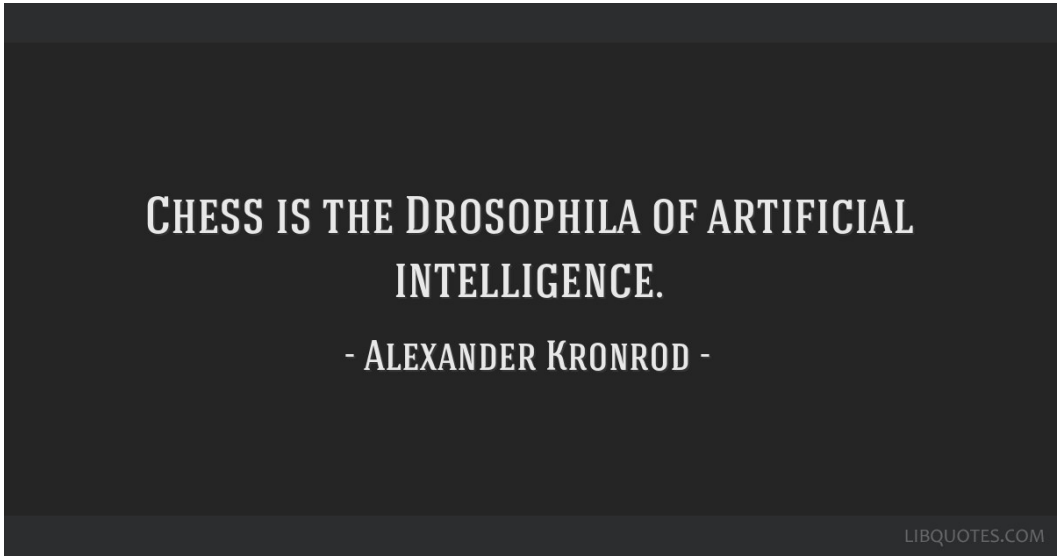
The game of checkers has recently been programmed into a general-purpose computer, using a "general prin-

C. E. Shannon, Computers and Automata, Proc. of IRE, vol. 41, pp. 1234-1241, 1953



- One way that's kind of a fun analogy to try to get some idea of what we're doing here to try to understand nature is to imagine that the gods are playing some great game like chess. Let's say a chess game. And you don't know the rules of the game, but you're allowed to look at the board from time to time, in a little corner, perhaps. And from these observations, you try to figure out what the rules are of the game, what [are] the rules of the pieces moving.
- You might discover after a bit, for example, that when there's only one bishop around on the board, that the bishop maintains its color. Later on you might discover the law for the bishop is that it moves on a diagonal, which would explain the law that you understood before, that it maintains its color. And that would be analogous we discover one law and later find a deeper understanding of it.

https://en.chessbase.com/post/feynman-using-chess-to-explain-science



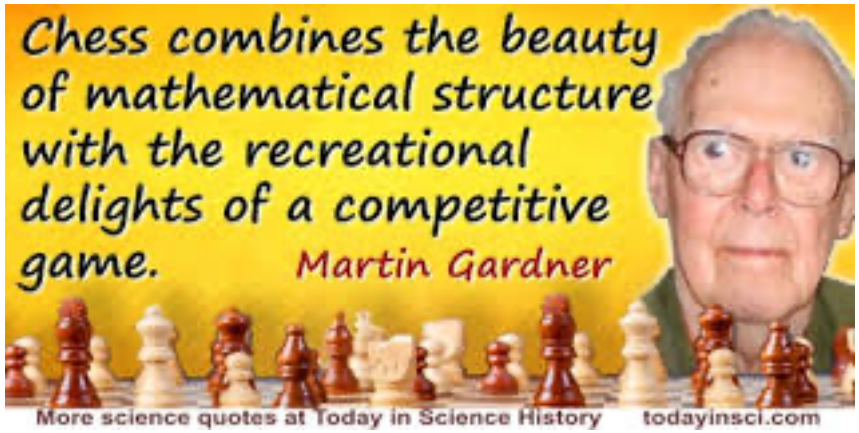
Aleksandr S. Kronrod

- This lecture introduces machine learning ideas that *power neural networks and deep learning*
- How do you teach a machine to play an AI game of strategy ?
- Explore the art of *trial-and-error* strategy and *reinforcement learning*
- Explore how AI, initially only knowing the rules, can *automatically search for the right path (sequence of moves)*
- Machine learning as the procedure to *automate the process* of retention of probable winning strategies and elimination of unsuccessful ones

http://jmc.stanford.edu/articles/drosophila/drosophila.pdf

https://en.wikipedia.org/wiki/Alexander_Kronrod

Hexapawn: The Drosophila of Machine Learning



Hexapawn Game

- Know just the rules. Do NOT try to analyze it first!

Start with a Beginner's mind!

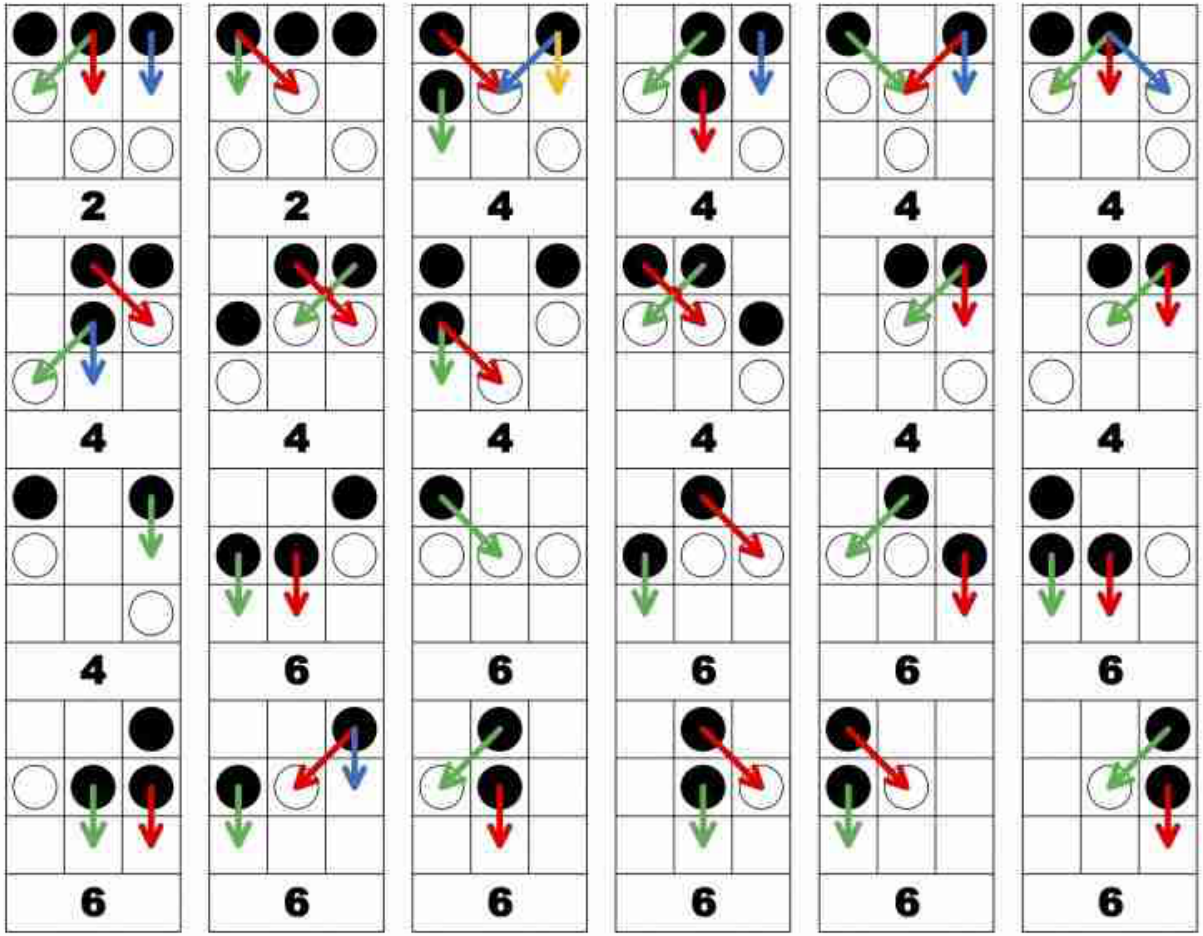
Feasible Moves

- Pawns can move one square forward into an empty square.
- Pawns can capture an opponent's pawn by moving forward diagonally.

When A Player Wins (Checkmate!)

- Get one of your pawns to the other side of the board or;
- Prevent the opponent from moving (e.g., no more pawn left)

Activity: Teach the Machine to play Hexapawn!



How many games to train the machine into a skillful player?

Are there ways to train it faster?

Machine Learning in AI Games

- Training Bias: Notice you have created a human-generated dataset that has a human bias
- How long did you take to ‘teach’ the machine? Conversely, how long did *you* ‘learn’ the game?
- Can you ‘*unlearn’*? Do you remember the mistakes made in your previous moves?
- Can a machine remove the human bias in the training? How about pit a machine against another machine?
- When you *teach* something, you *learn it twice* - Joseph Joubert

Activity: Fish-Flavored Lollipops

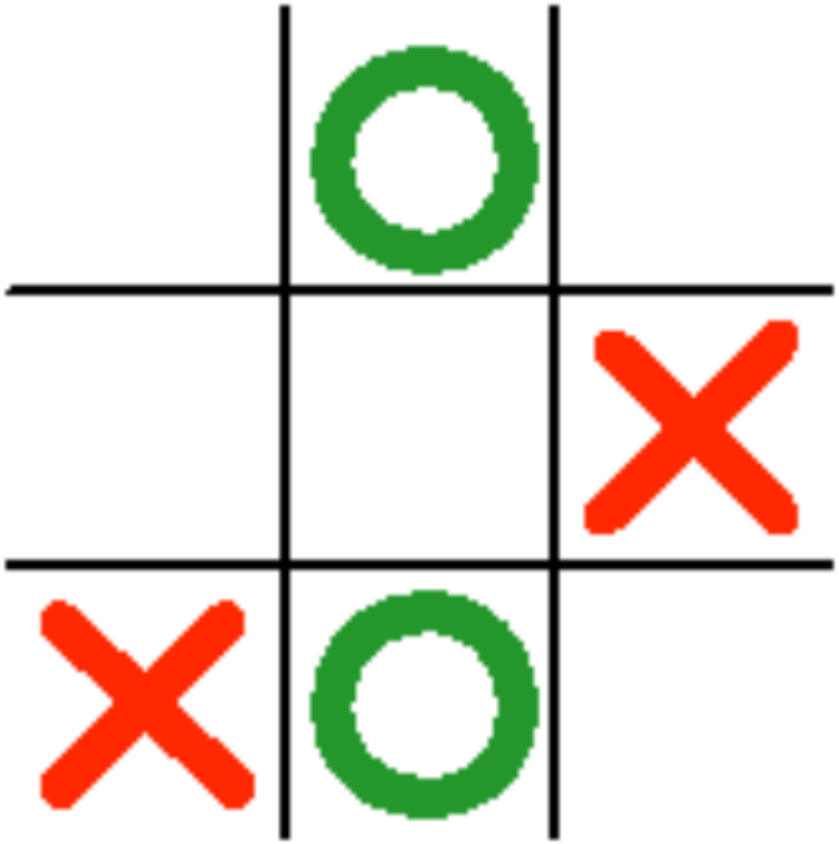
- Train Nemobot to play NIM Game skilfully!

Fish-Flavored Lollipops is a variant of Nim, an ancient math puzzle. When the game starts, I will show you 13 lollipops, where the last one of them is fish-flavored. It tastes so disgusting that nobody wants to eat it. The lollipops will be placed in one line, and you and I will take turns to take lollipops from the row. You can't take more than 3 lollipops at a time, and you can't skip your turn. Whoever takes the last lollipop (the fish-flavored one) lose the game.

Stuart C. Hight, director of research studies at Bell Labs in Whippany, NJ, built a machine called NIMBLE (Nim Box Logic Machine) in 1960s.

Activity: Tic-Tac-Toe

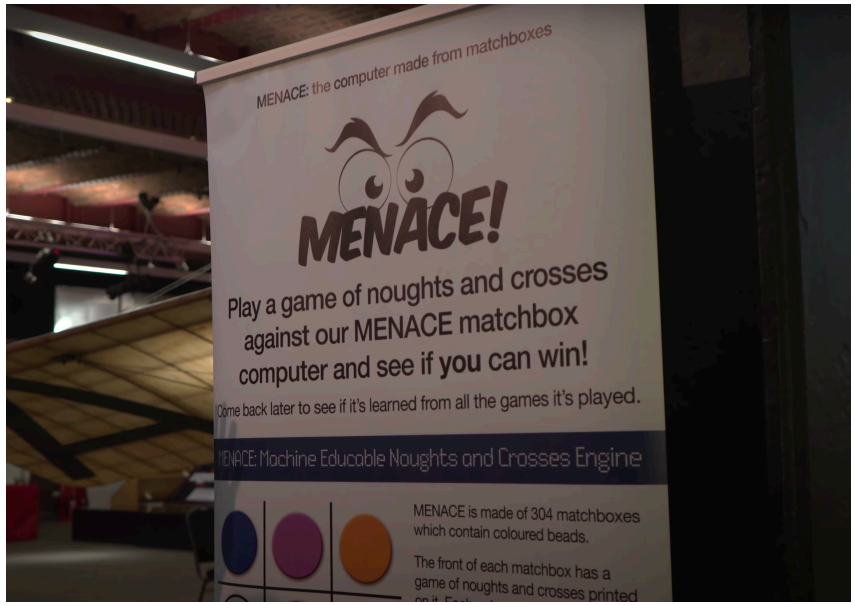
- Train the machine to play Tic-Tac-Toe skilfully
- The original drosophila by Donald Michie
- MENACE: a Machine Educable Noughts And Crosses Engine using 304 matchboxes
- Train your machine here: https://www.mscroggs.co.uk/menace



MENACE Model

MENACE Model

- MENACE: Machine Educable Noughts And Crosses Engine
- Created by researcher Donald Michie in 1961
- Use 304 matchboxes to play Tic-Tac-Toe
- Try it here: `https://www.mscroggs.co.uk/menace/`



Use MENACE to play Tic-Tac-Toe

All layouts that were rotated versions of the same thing or that were symmetrical to each other with a single matchbox.

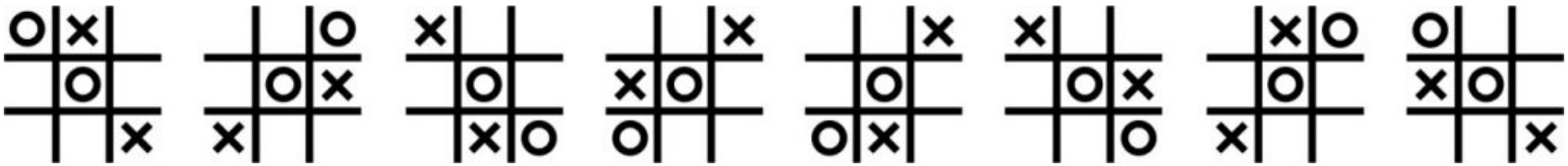


Figure: One box would represent all the layouts

Build your MENACE model:

- Each matchbox represents a specific board layout of tic-tac-toe.
- A range of coloured beads are placed in each matchbox. Each colour corresponds to a possible move which MENACE could make from that position.

[1] https://opendatascience.com/menace-donald-michie-tic-tac-toe-machine-learning/

Use MENACE to play Tic-Tac-Toe

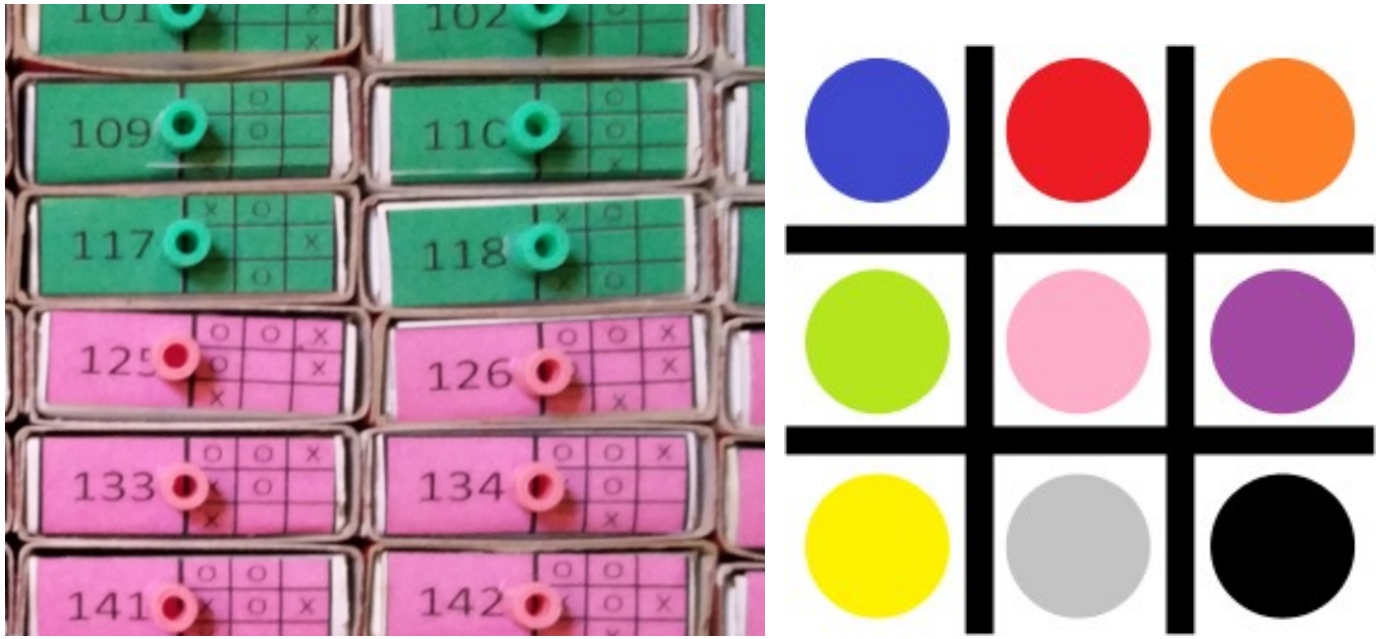


Figure: When training begins, all boxes contain colour-coded beads, where each colour represents a move (or position) on a board.

Use MENACE to play Tic-Tac-Toe

Train your MENACE model:

- MENACE makes a move when the human player randomly picks a bead out of the box that represents the game's current state. The colour of the bead determines where MENACE will move.
- At the end of each game, if MENACE loses, each bead MENACE used is removed from each box. If MENACE wins, three beads the same as the colour used during each individual turn are added to their respective box. If the game resulted in a draw, one bead is added.
- After 200 games play out, the matchboxes will have more of some beads than of others. This makes it more (or less) likely for a given bead to be picked.

Use MENACE to play Tic-Tac-Toe

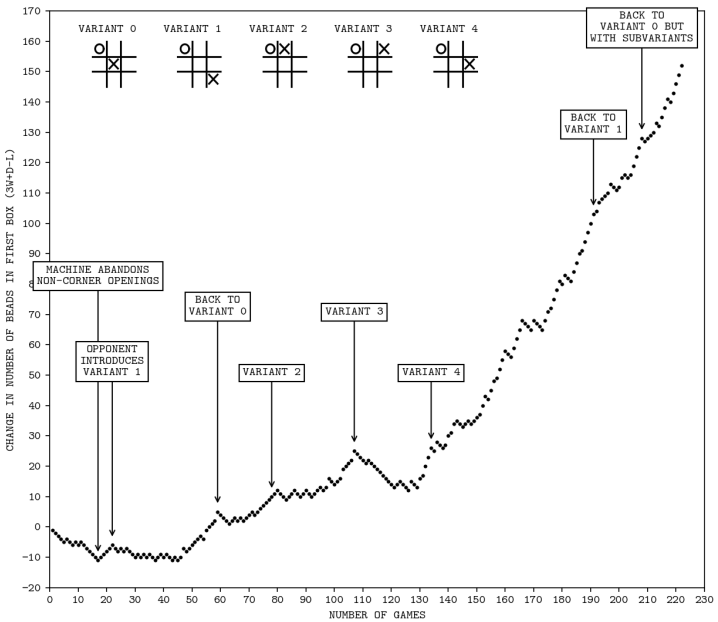


Figure: The result when MENACE plays against a perfect computer

- A draw is considered positive, because it suggests MENACE has learned.
- When MENACE played a with a random picking opponent, the result is a near-perfect positive correlation.
- Try it here: `https://www.mscroggs.co.uk/menace/`

[1] `https://www.mscroggs.co.uk/blog/19`

Nim Game

Nim is an ancient game with many variations. In this case we'll use one of its simplest forms: Lay out ten pencils (or coins, buttons, etc.) in a row on the table. Two players take turns removing either one, two or three pencils. The player that takes the last pencil loses.

[1] https://www.i-am.ai/build-your-own-ai.html

Use MENACE to Play Nim

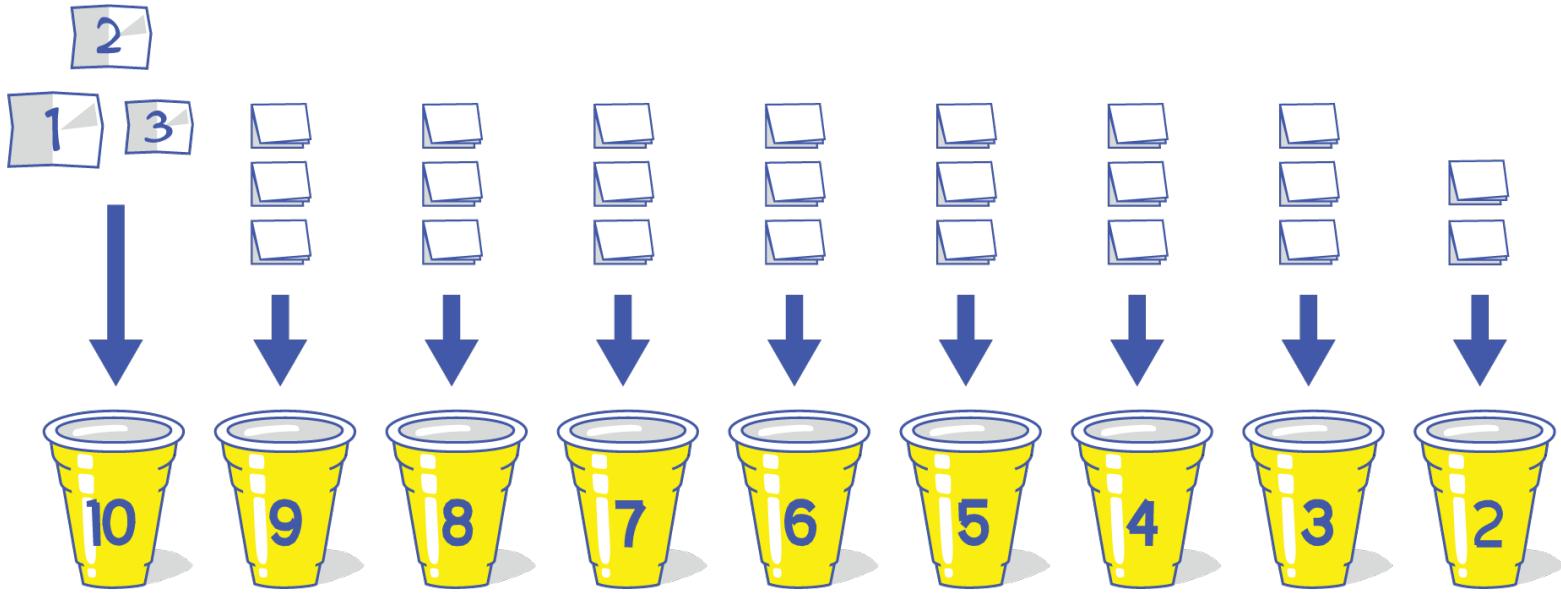


Figure: Build an artificial intelligence that can play Nim

Build your AI (MENACE model):

- Step 1: Get nine cups, label them with the numbers 10 through 2 and place them in a row in front of you.
- Step 2: Cut three small square pieces of paper of approximately the same size and write the numbers 1, 2 and 3 in them.

Using MENACE to Play Nim

- Step 3: Fold each piece of paper in four and place them on cup number 10.
- Step 4: Repeat step 2 and step 3 until each cup has three pieces of paper numbered 1, 2, 3 inside. The last cup, number 2, only needs two pieces of paper with the numbers 1 and 2.

Now your AI is able to play Nim against you. Decide who goes first. When it's the AI's turn to play, count how many pencils are left and grab the cup with that number. Without looking, pick a piece of paper from inside, the number written on it is the number of pencils that the AI takes. Put the piece of paper back in the cup.

Using MENACE to Play Nim

Train your AI (MENACE model):

Take turns with the AI being the first to play. Whenever you pick a piece of paper for the AI leave it next to the cup it came from. When the game finishes do the following:

- If the AI won, re-fold the papers and put them back inside each cup.
- If the AI lost, discard the piece of paper that made it take its last action instead of putting it back in the cup. The AI learned this was a bad choice, so it won't repeat it. Fold the rest of the papers and put them inside their cups.
- However, if a cup only has one piece of paper in it, put it back and discard the piece of paper from the previous action instead. Cups should always have at least one piece of paper.

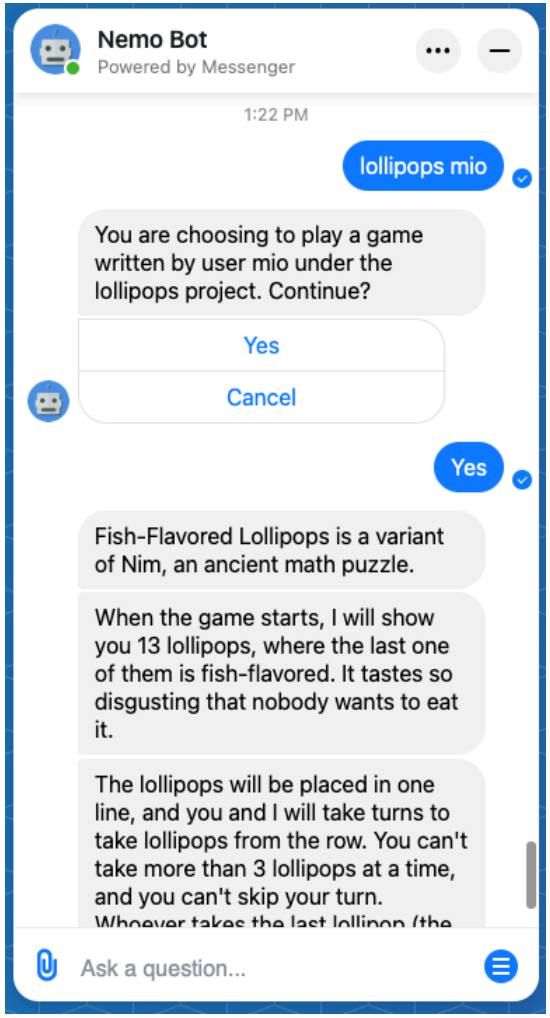
After playing enough times, each cup will end with only one piece of paper: the perfect move in that situation.

Nim Game on Nemobot

Now you can try the Nim game from the Nemobot.

Instructions:

- Send "lollipops mio" to Nemobot to start the game.
- Open `http://algebragamification.com:3032/` to check the result when AI plays against all nemobot users.



Machine Learning in AI Games

- Donald Michie invented reinforcement learning as a form of statistical learning in 1961
- Reinforcement Learning focuses on how to reward and penalize strategy through experience
- Is it true that a more complex game requires more training (i.e., a bigger training data set)?
- Will the learning be slower for more complex game?
- What is the best strategy to rewarding and penalization in Reinforcement Learning?
- Is Backpropagation an optimal strategy?

Machine Learning in AI Games

- Alan Turing in his 1948 paper “Intelligent machinery” (https://weightagnostic.github.io/papers/turing1948.pdf) introduces the idea of rewarding and penalizing strategy through experience – early form of Reinforcement Learning
- Is it true that a more complex game requires more training (i.e., a bigger training data set)?
- But is a *bigger* training data set always better?
- Will the learning be slower for more complex game, e.g., checker, chess, Go?
- From Codebreaking to Computing: Remembrances of Bletchley Park 50 Years Later

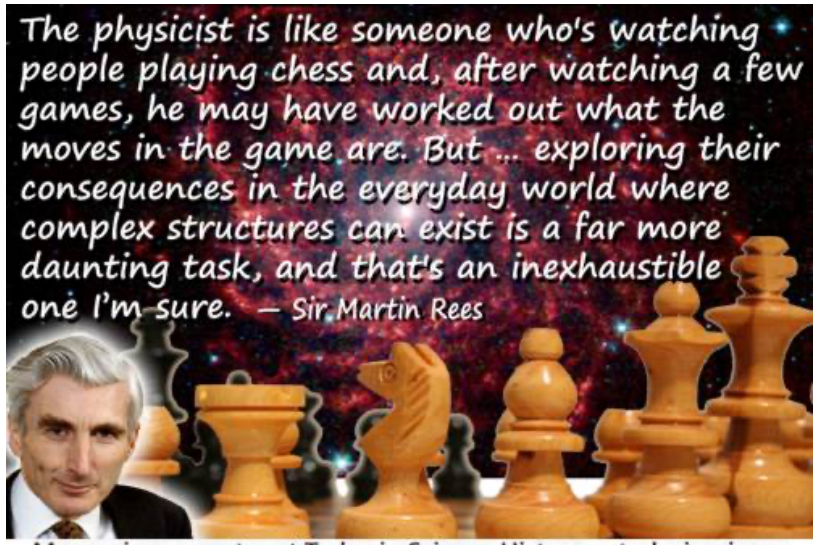
https://www.computer.org/csdl/magazine/an/1995/01/man1995010032/13rRUxYIN6g

Summary of Machine Learning

- Game-learning machines playing Hexapawn, Nim and Tic-Tac-Toe serve as the ideal *drosophila* of machine learning in AI
- Machine learning ideas:
  - Supervised Learning
  - Training
  - Trial-and-Error Strategy
  - Reinforcement Learning by Reward and Penalization
  - Backpropagation

"When you ask what are electrons and protons I ought to answer that this question is not a profitable one to ask and does not really have a meaning. The important thing about electrons and protons is not what they are but how they behave, how they move. I can describe the situation by comparing it to the game of chess. In chess, we have various chessmen, kings, knights, pawns and so on. If you ask what chessman is, the answer would be that it is a piece of wood, or a piece of ivory, or perhaps just a sign written on paper, or anything whatever. It does not matter. Each chessman has a characteristic way of moving and this is all that matters about it. The whole game os chess follows from this way of moving the various chessmen."

– Paul A.M. Dirac



https://www.quantamagazine.org/martin-rees-on-the-future-of-science-and-humanity-20181205