Lecture 7

Cezar Ionescu
29/06/2019

DEPARTMENT FOR
CONTINUING
EDUCATION

UNIVERSITY OF
OXFORD

Administrative

- Homework from 22/06/2019 due now!
- Please complete and hand in the declarations of authorship.

Questions?

Solution to homework from lecture 6

Games and AI

Progress in AI has often been connected to game-playing systems.

- 1956: Checkers – Arthur Samuel
- 1992: TD-Gammon – Gerald Tesauro
- 1997: *Deep Blue* (IBM) defeats Kasparov
- 2016: AlphaGo (DeepMind, Google) defeats Lee Sedol

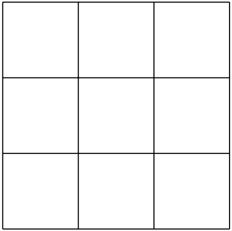
MENACE

In 1960, Donald Michie invented MENACE (*Matchbox Educable Noughts And Crosses Engine*). MENACE is an automated system that learns to play optimal tic-tac-toe (noughts and crosses) by playing against itself. It is built out of matchboxes, bits of paper, and sugar drops.

Figure 1: Donald Michie

Figure 2: “Cip” sugar drops

Noughts and crosses



Noughts and crosses

	X	

Noughts and crosses

O		
	X	

Noughts and crosses

O		X
	X	

Noughts and crosses

O		X
	X	
O		

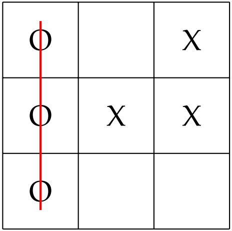
Noughts and crosses

O		X
	X	X
O		

Noughts and crosses

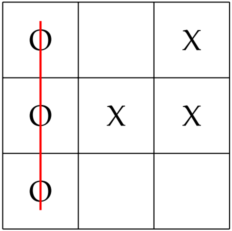
O		X
O	X	X
O		

Noughts and crosses



O		X
O	X	X
O		

Noughts and crosses



O		X
O	X	X
O		

Winner: O

Building a version of MENACE

Our version looked similar to the one built by James Bridle:

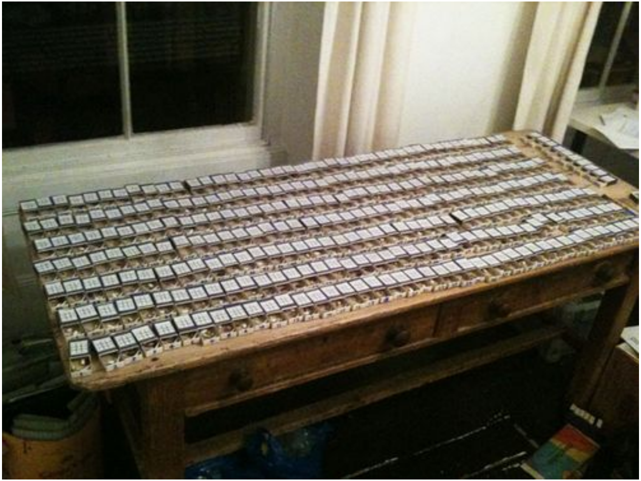


Figure 3: MENACE

Building MENACE

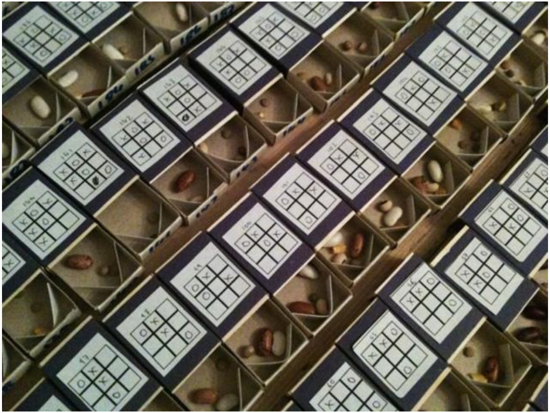
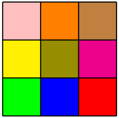


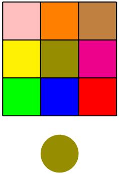
Figure 4: Close-up

…but we had much smaller sugar drops and were colour-coding the empty slots.

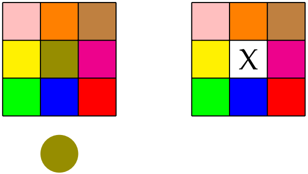
MENACE in action



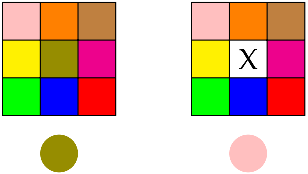
MENACE in action



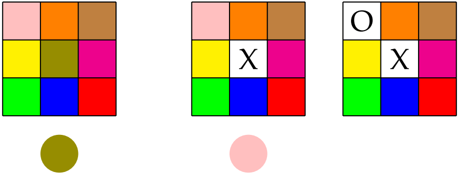
MENACE in action



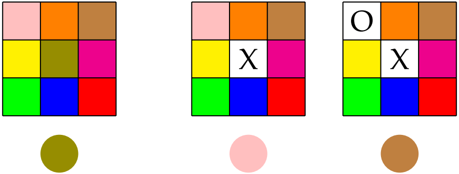
MENACE in action



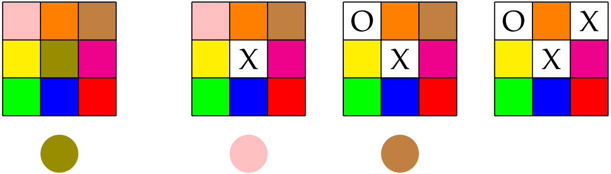
MENACE in action



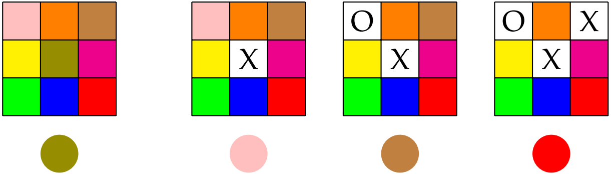
MENACE in action



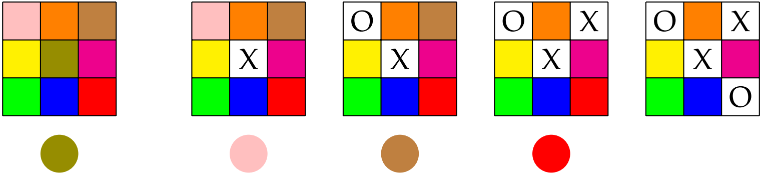
MENACE in action



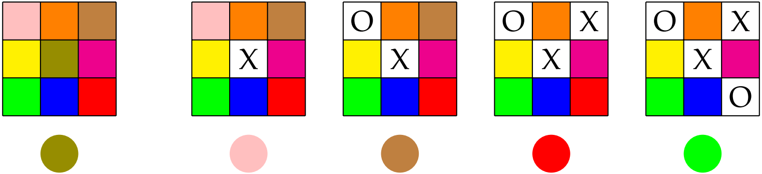
MENACE in action



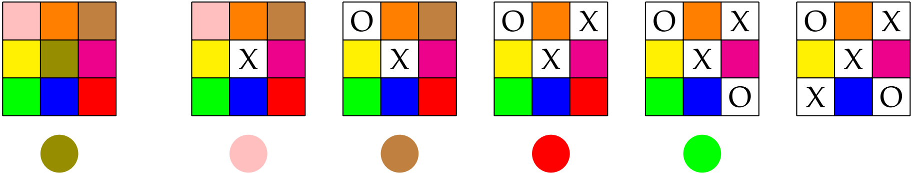
MENACE in action



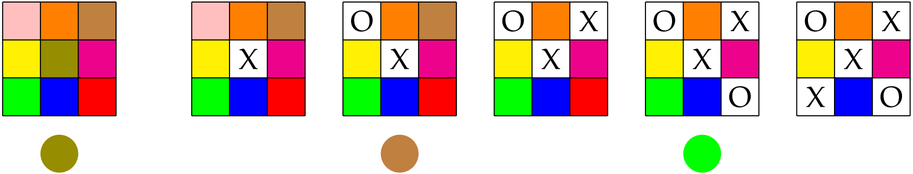
MENACE in action



MENACE in action



MENACE in action



Building MENACE

There are about `3000` possible boards. However, due to the symmetry of the board, it is possible to reduce this number by a factor of ten. This is a form of *knowledge engineering*: using available information to reduce the complexity of the learning system.

Policies

What does the matchbox system represent? What is the input to the matchbox system? What is the output?

Policies

What does the matchbox system represent? What is the input to the matchbox system? What is the output?

```
matchboxSystem : Board -> Move
```

The matchbox system tells us what action to take in a given state. Such a function is called a *policy*.

Policies

Examples of policies:

- deciding how to turn the steering wheel and act on the pedals in a self-driving car
- deciding how to control a robot arm in order to catch a ball
- deciding how to set emissions barriers in order to reduce greenhouse gases without damaging the economy
- …

In all these examples, actions *now* have consequences *in the future*. A policy must have "foresight".

Policies

```
policy : State -> Action
```

- tic-tac-toe: `State = Board`, `Action = Move`
- self-driving car: `State = Sensors`, `Action = (WheelTurn, BrakePress)`
- robot arm: `State = (BallPos, HandPos)`, `Action = (ArmTurn, FingerMove)`
- emissions barriers: `State = (EconomyVars, ClimateVars)`, `Action = EmissionLevel`
- …

Policies

A “good” policy attains a long-term goal, not just an immediate one.

- Tic-tac-toe:
    - Example of a long-term goal in tic-tac-toe: maximise the number of wins in the long run (or minimise the number of losses).
    - Example of a short-term goal in tic-tac-toe: occupy the centre.

Policies

- Self-driving cars:
  - Example of a long-term goal in self-driving cars: smooth driving, minimise consumption, reach destination quickly.
  - Example of a short-term goal in self-driving cars: turn left to avoid pedestrian.

Policies

- Robot arm:
  - Example of a long-term goal in controlling a robot arm: catch the ball.
  - Example of a short-term goal in controlling a robot arm: bend the arm.

Policies

- Setting emissions:
  - Example of a long-term goal in setting emissions: keep global warming below 2°C.
  - Example of a short-term goal in setting emissions: eliminate diesel cars.

Matchboxes

If the matchbox system represents a policy, what does a single matchbox represent?

```
matchbox : ? -> ?
```

Matchboxes

If the matchbox system represents a policy, what does a single matchbox represent?

```
matchbox : ? -> ?
```

```
matchbox b : (Board, Move) -> Number
```

A matchbox represents the system's current valuation for the various possible moves. Bigger values correspond to better moves. The matchboxes implement a value function:

```
value : (Board, Move) -> Number
```

Matchboxes

The equivalent of matchboxes in self-driving cars: a function that gives the value of various possible actions depending on the sensor readings.

The equivalent of matchboxes in controlling a robot arm: a function that gives the value of various possible arm movements depending on the position of the ball and of the hand.

The equivalent of matchboxes in setting emissions: a function that gives the value of possible emission barriers depending on the state of the climate and the state of the economy.

State-action value function

```
value : (State, Action) -> Number
```

In self-driving cars:

```
value : (Sensors, (WheelTurn, BrakePress)) -> Number
```

For robot arm:

```
value : ((BallPos, HandPos), (ArmTurn, FingerMove)) -> Number
```

For emissions barriers:

```
value : ((EconomyVars, ClimateVars), EmissionLevel) -> Number
```

Function approximation

In each of these cases, knowledge of this function would imply the solution to the problem. If we already know the values of many `(State, Action)` pairs, we can use a neural network to approximate the state-action value function. Neural networks are good *function approximators*.

Reinforcement learning

In many cases, however, we do not have known values for `(State, Action)` pairs. The state-action value function must be approximated by *trial and error*. The intuition is that of *behavioural conditioning*. *Reinforcement learning* is a way of implementing efficient trial and error behaviour. The algorithm implemented by MENACE is the simplest example of reinforcement learning.

Choosing a move

In MENACE, we choose a move stochastically. Moves with better values (more representatives in the matchbox) have better chances of being chosen.

```
choose : Matchbox ~> Move
```

Why not just choose the move that has the maximal number of represenitives?

Credit assignment

When X or O lose a game, all moves that lead to that loss are penalised. It is not always easy to assign credit or blame to the individual moves. Good moves lead to good results *on average*, not necessarily every time. Choosing stochastically reduces the risks of throwing away good moves.

Exploration versus exploitation

Donald Michie’s original version added sugar drops to winning moves. Realistic examples follow this pattern and “reward” winning moves. This raises a problem similar to that of credit assignment: we might reward average, or even bad moves. We want to get the rewards promised by well-valued moves, but we risk missing even greater rewards if we do not try out new moves from time to time. This is known as the problem of “exploration versus exploitation”.

Summary

MENACE illustrates the most important problems and methods of AI:

- knowledge engineering
- credit assignment
- exploration vs exploitation (cf. if we had only one sugar drop per move)
- randomisation (cf. outputing typewritten vs probability of typewritten)
- function approximation
- reinforcement learning

Homework

A smart traffic control system can adjust the delays of traffic lights in order to alleviate bottlenecks in the flow of vehicles and ensure a smooth driving experience. In order to achieve this, the system must anticipate how the delays will influence the evolution of the traffic pattern. We want to train this system using an idealised version of the matchbox system.

- What are the states? What are the actions (sugar drops)?
- What is the equivalent of a matchbox for this system?
- What could be a long-term goal for this system? What could be a short-term goal of the system?
- How would the updating process work (how can bad actions be eliminated)?