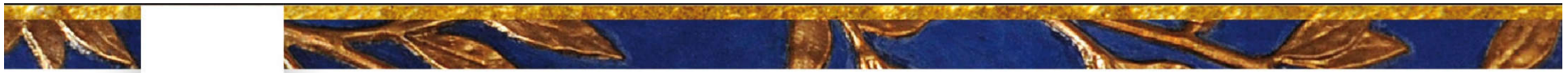


UPPSALA
UNIVERSITET

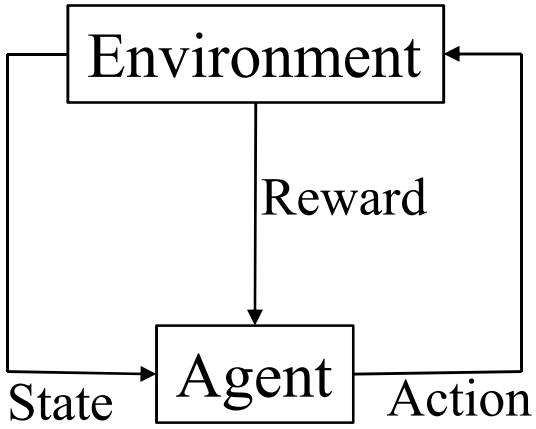
Natural Computation Methods in Machine Learning (NCML)

Lecture 7: Reinforcement Learning

UPPSALA UNIVERSITET

Reinforcement Learning (RL)

- Put the neural networks (ANNs) aside for a while
  - RL is not about ANNs, it is a field in its own right
  - (ANNs are often used in RL, though)
- Learning by interaction with an environment, to maximize som long-term scalar value
  - (or to minimize a cost)
  - Learning by trial-and-error
- Applicable to a very large set of real world problems
- Strongly related to Markov Decision Processes and Dynamic Programming

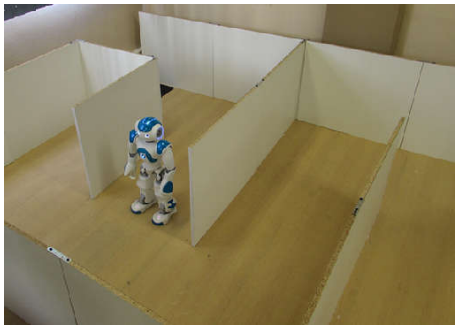


UPPSALA UNIVERSITET

Reinforcement Learning (RL)

Application examples

- Board games
  - Checkers (1959), Tic-Tac-Toe (1961), Backgammon (1992), Go (2015) ...
- Robot navigation (maze solving in general)
- Pole balancing (control of unstable systems)
- Scheduling (elevators, schools, airline crews, ...)
- Traffic control (cars, trains, network routing, ...)

UPPSALA
UNIVERSITET

MENACE

Machine Educable Noughts and Crosses Engine

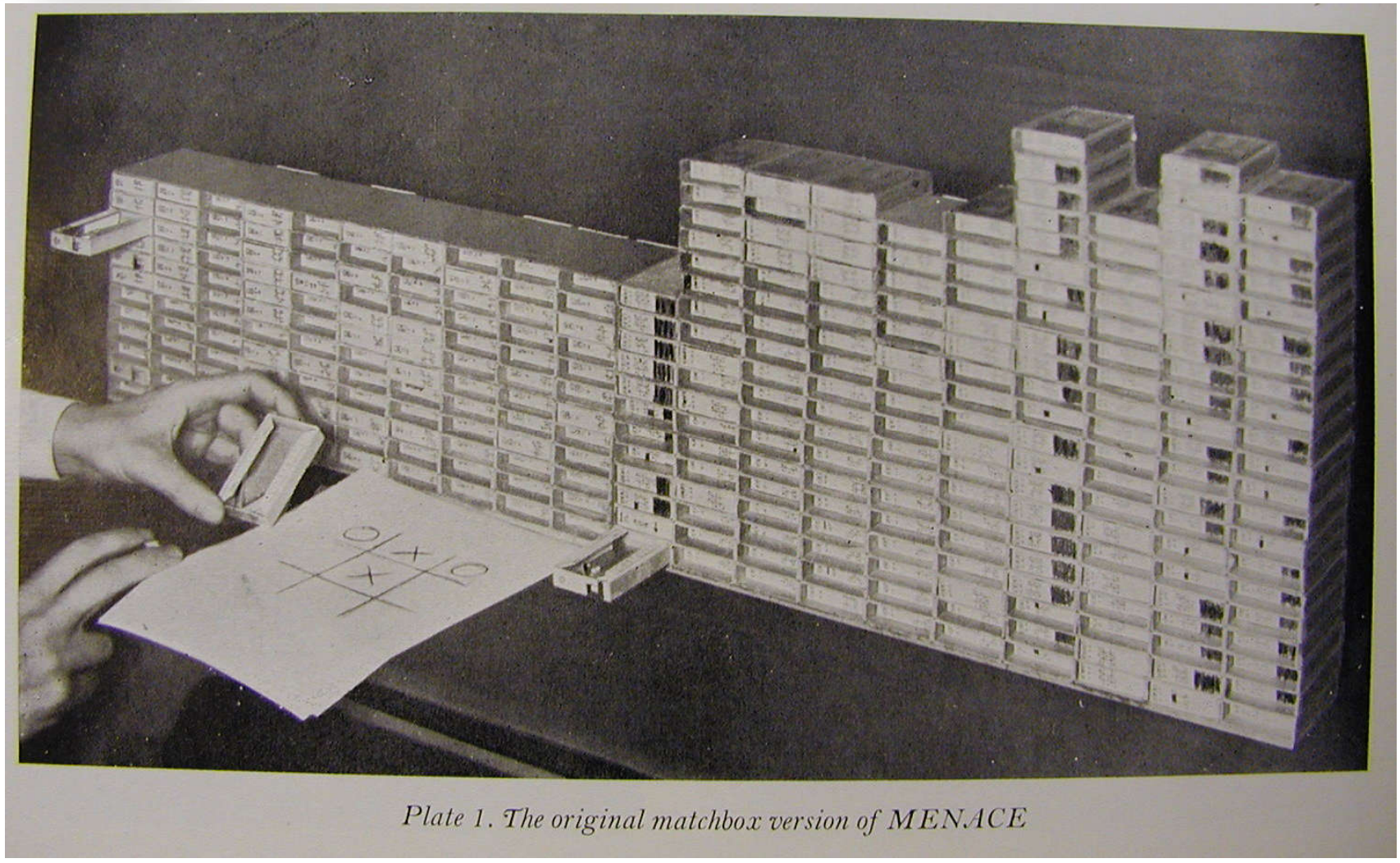


Plate 1. The original matchbox version of MENACE

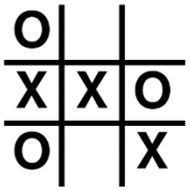
Donald Michie, 1961

UPPSALA UNIVERSITET

MENACE

Machine Educable Noughts and Crosses Engine

- Learns to play Tic-Tac-Toe (perfectly)
- The game board has at most $3^9 = 19683$ states
  - only 304, if we remove illegal and symmetric states
- One matchbox for each board position (state)
- Fill each box with a random number of coloured beads
  - 9 colours, corresponding to the 9 possible moves (actions)



+

+

=

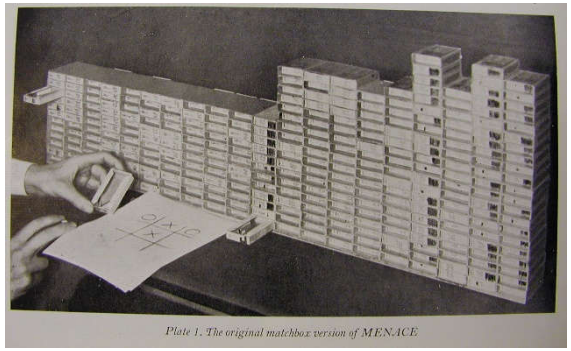



Plate 1. The original matchbox version of MENACE

UPPSALA UNIVERSITET

MENACE

How it works

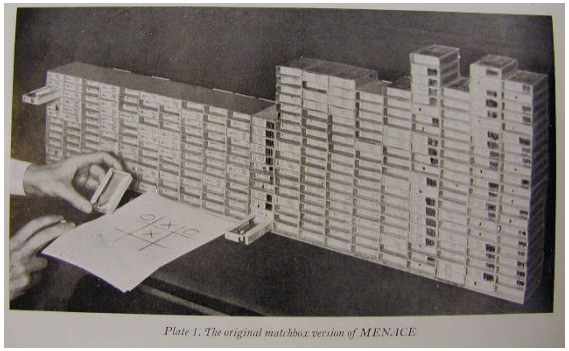
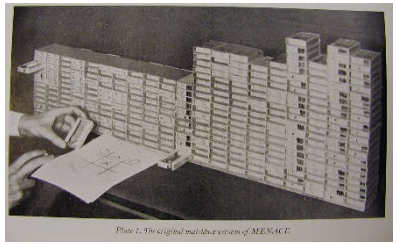


Plate 1. The original matchbox version of MENACE

- Given a board position (state)
  - Select the corresponding box
  - Draw random bead from the box
  - Colour decides position (the action)
  - Make the move and leave the bead outside the box
- When game ends
  - If MENACE won, put each bead back in its box + one more of same colour
  - If it lost, throw away the drawn beads
  - If the game was a draw, reset (put each bead back)
- It took about 150 games for MENACE to learn the optimal strategy, when playing against D. Michie

UPPSALA UNIVERSITET

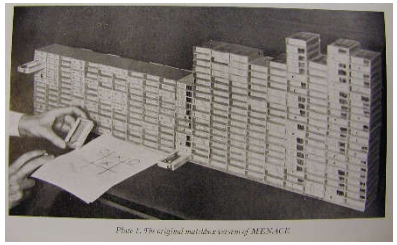


MENACE

Some notes

- All beads of a certain colour may be removed from a box!
  - That move will never be selected again
  - Neither would you, if you lost that many games ...
- A box may be emptied completely!
  - MENACE will refuse to play, if coming to that state
  - So would you, if you lost that many games ...
- That empty box may represent the very first state!
  - MENACE will refuse to play at all
  - So would you, if you lost that many games ...
- Game rules can be (and was) given as a prior
  - In initialization, make sure illegal moves have no beads
  - The machine can focus, not on *how* to play, but how to play *well*
- How to play against the machine? (on what level)

UPPSALA UNIVERSITET

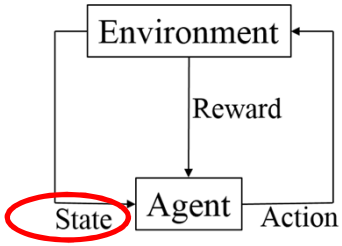


MENACE

Some notes

- MENACE gets no feedback until the game ends
  - Feedback depends on a sequence of moves
  - Difficult to decide which moves in the sequence were the most significant for the result
  - A credit assignment problem! (here, a temporal one)
  - Here, all moves in sequence are blamed/credited
- Are all moves blamed/credited by the same amount?
  - Not necessarily, depends on how boxes are seeded
  - Boxes representing late game states should have fewer beads of each colour

UPPSALA UNIVERSITET



RL concepts and issues

State

- The State is a description of the environment's state
  - In MENACE, the board position
- Usually not a complete description of the environment, ...
  - Think of it as a vector of sensor values to a robot
- but must contain enough information for the agent to solve its task!
  - If not, the mapping (state→action) is no longer a function! (many-to-many instead of many-to-one)
    - Very common reason for experiment failures

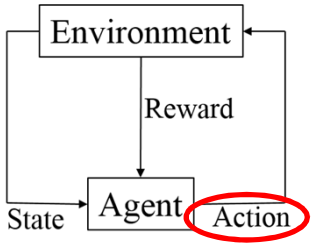
UPPSALA UNIVERSITET

RL concepts and issues

Reward (reinforcement signal, grade)

- The reward is quality measure of the environment's state
  - Only indirectly, that of the agent's
    - The environment may have been affected by other things as well – e.g. random events or other agents
    - The agent's state may be incomplete
    - It does not say which actions in sequence to blame (or which other actions would have been better)
- This, very noisy, feedback is what we try to maximize (in the long run)
- Tip: Don't assume that negative rewards must be bad and positive rewards must be good!
  - To the agent, the reward is just a scalar to be maximized
  - That maximum may very well be negative (or zero)

UPPSALA UNIVERSITET



RL concepts and issues

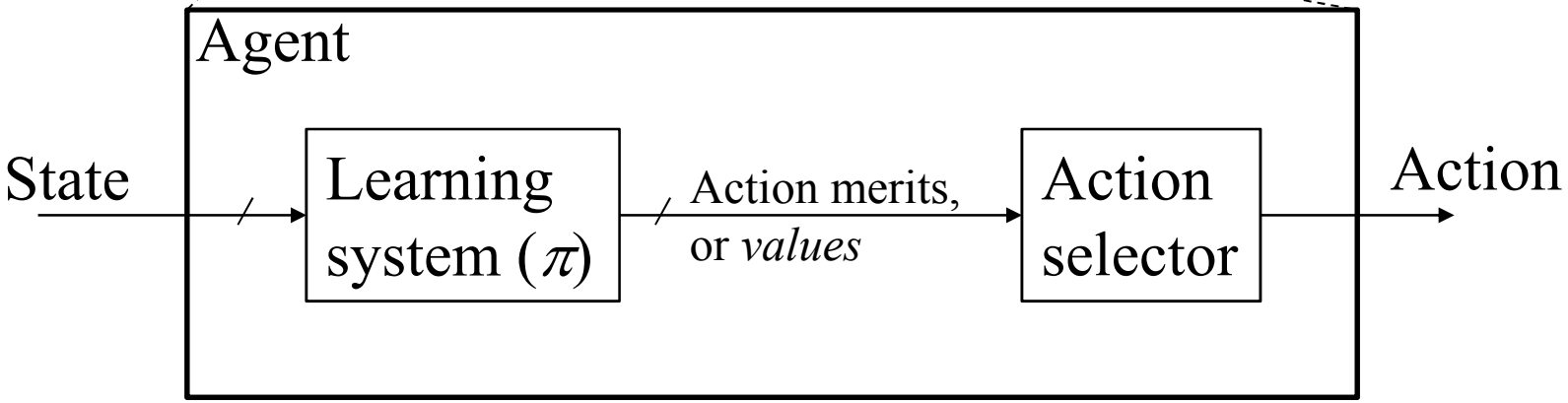
Action

- Actions affect the environment and, indirectly, the next state/reward
  - The environment may be affected by other things as well
- Actions can be discrete or continuous
  - discrete: finite number of possible actions per state
    - as in MENACE
    - The agent selects an action, i.e. it is a classifier
  - continuous: infinite number of possible actions
    - for example, controlling the steering wheel of a car
    - The agent is a continuous function approximator
- For now, we will assume that they are discrete

UPPSALA UNIVERSITET

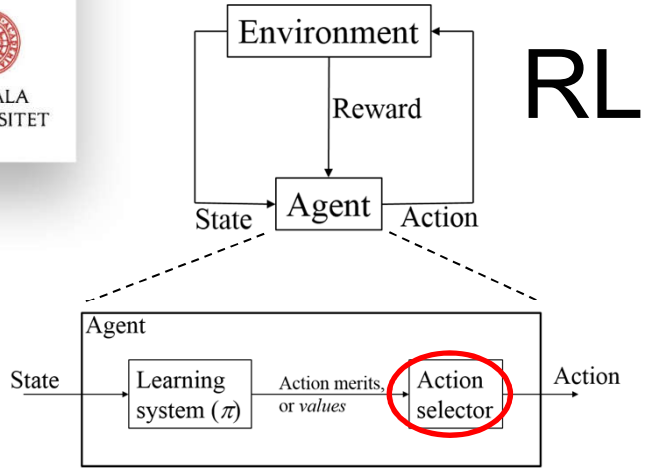
RL concepts and issues

The Agent



- Action selection must be stochastic!
  - The reward does not say which other actions, than the ones selected, would have been better
  - The agent must *explore*, to find better solutions
  - To explore = to sometimes select actions which (at that time) seem to be sub-optimal

UPPSALA UNIVERSITET



RL concepts and issues

Action Selection

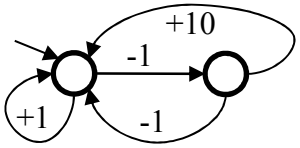
- Epsilon-greedy (ε-greedy) action selection
  - With probability ε, select a random action
    - i.e. ignore the merit values from the learning system
  - Otherwise be *greedy*
    - *Greedy selection* is to choose the action which is currently believed to be the best (has highest merit value)
- Bolzmann selection (roulette-wheel sampling)
  - Select stochastically, proportionally to merit value
  - MENACE does this, in effect

UPPSALA UNIVERSITET

RL concepts and issues

Delayed v.s. Immediate RL

- Rewards should be associated with a sequence of actions (*delayed* RL), not just the latest one (*immediate* RL)
  - MENACE implements delayed RL, since it blames/credits all moves
- Also problems where rewards are payed after every action, may require delayed RL!
  - Simple example:



What's the optimal strategy here?
Is this strategy discoverable by immediate RL?

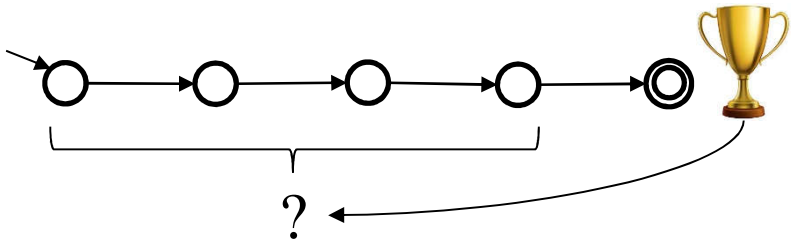
  - Real world example: Traffic control
  - Better to think of rewards as always being payed out
    - It's just that they are sometimes 0

UPPSALA UNIVERSITET

RL concepts and issues

The Temporal Credit Assignment problem

- In delayed RL, how to decide which actions in a sequence to give credit/blame for the result

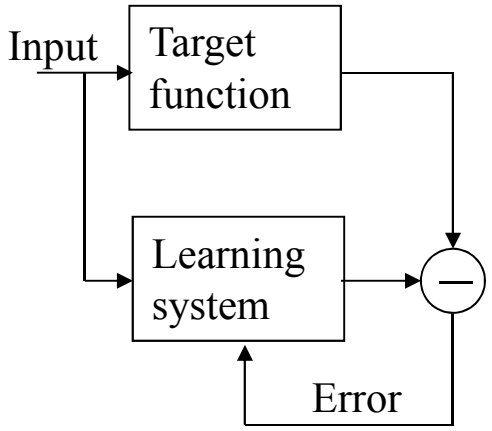


- Common solution: Blame late decisions more than earlier ones
  - as MENACE does, if the boxes are seeded properly
  - not necessarily the best way, but easy to automate (and we usually don't know enough about the problem to be more clever)

UPPSALA UNIVERSITET

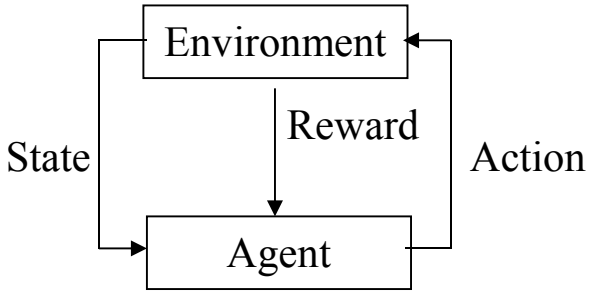
Comparison to supervised learning

Supervised Learning (SL)



- SL <u>imitates</u>
  - and can therefore never exceed the performance of its 'teacher'
- SL does not (usually) interact
- Feedback is specific and instructive
  - it says exactly what we want from the outputs

Reinforcement Learning (RL)



- RL <u>explores</u>
  - and therefore can outperform its 'teacher'
- RL <u>interacts</u>
- Feedback in RL is noisy, indirect and non-specific
  - It does not say which other actions would have been better

The two can be combined!

UPPSALA UNIVERSITET

More RL concepts

Goal, Value, and Policy

- The goal is to maximize the long-term reward
  - but we will soon reformulate this
- Sum of (expected) future rewards? (the *Return)*

$$R_t = E\left(\sum_{k=0}^{h} r_{t+k}\right)$$

- The Return has a *finite horizon* (h)
  - would like to have infinite horizon ($h{=}\infty$)
  - but that would make the sum infinite
    - not good since we try to maximize it
- Solution: Maximize a discounted sum instead

UPPSALA UNIVERSITET

More RL concepts

Goal, Value, and Policy

- Define the *Value* at time t, as a discounted sum of expected future rewards:

$$V_t = E\left(\sum_{k=0}^{\infty} \gamma^k r_{t+k}\right)$$

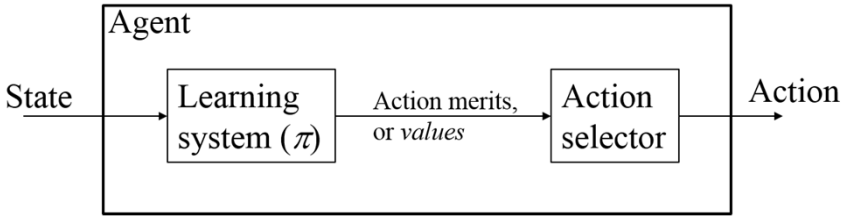
- γ is a *discount factor*, $0 \leq \gamma < 1$
  - side effect: decides short/long sightedness
  - Rule of thumb: The agent looks $\frac{1}{1-\gamma}$ time steps ahead
- Geometric sum ➔ finite (as long as $\gamma < 1$)
- Continuous variants exists
  - An integral instead of a sum

UPPSALA UNIVERSITET

More RL concepts

Goal, Value, and Policy

- A *policy* is a function π from states to actions
  - i.e. it is the learning system in the agent



- The goal (reformulated) is to find an optimal policy, π^*, which for each state selects the action which maximizes V
  - This is a Markov Decision Problem, usually solved by some form of Dynamic Programming
  - RL (in this form) can be seen as a generalization of Dynamic Programming

UPPSALA UNIVERSITET

Challenges

- Implement MENACE as a table of values, and try to make it learn by playing against itself!
  - Nowadays, this should work even if you don't remove illegal/symmetric states
- How many matchboxes would we need if we wanted MENACE to play Connect Four instead?
- Does a reinforcement learning problem have to have an end state? (as in a board game)