RLearning:
Short guides to reinforcement learning

Unit 3-1: Overview Reinforcement Learning

Davud Rostam-Afschar (Uni Mannheim)

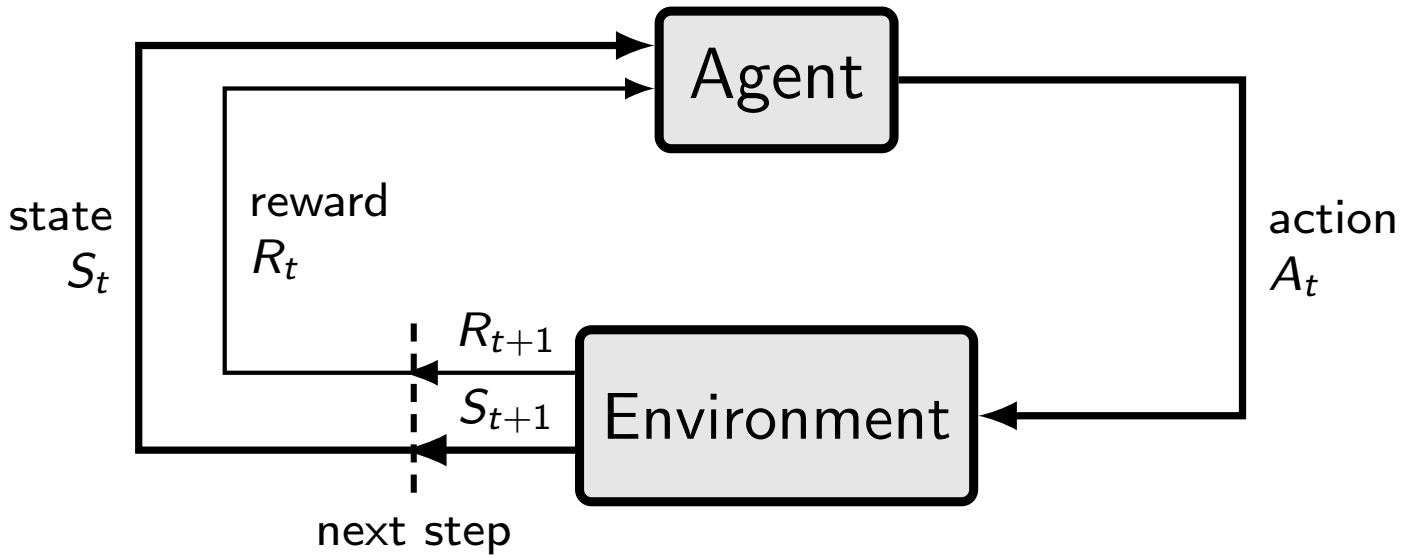
How computers (humans) learn?

Markov Decision Process

- Definition
  - States: $s \in S$
  - Actions: $a \in A$
  - Rewards: $r \in \mathbb{R}$
  - Transition model: $\mathbb{P}(s_t \mid s_{t-1}, a_{t-1})$
  - Reward model: $\mathbb{P}(r_t | s_t, a_t)$
  - Discount factor: $0 \leq \gamma \leq 1$
    - discounted: $\gamma < 1$
    - undiscounted: $\gamma = 1$
  - Horizon (i.e., # of time steps): h
    - Finite horizon: $h \in \mathbb{N}$
    - infinite horizon: $h = \infty$
- Goal: find optimal policy π^* such that

$$\pi^* = \operatorname*{argmax}_{\pi} \sum_{t=0}^{h} \gamma^t \mathbb{E}_{\pi} [r_t]$$

Reinforcement Learning Problem



Goal: Learn to choose actions that maximize rewards

Reinforcement Learning

- Definition
  - States: $s \in S$
  - Actions: $a \in A$
  - Rewards: $r \in \mathbb{R}$
  - Transition model: $\mathbb{P}(s_t \mid s_{t-1}, a_{t-1})$ — unknown
  - Reward model: $\mathbb{P}(r_t | s_t, a_t)$ — model
  - Discount factor: $0 \leq \gamma \leq 1$
    - discounted: $\gamma < 1$
    - undiscounted: $\gamma = 1$
  - Horizon (i.e., # of time steps): h
    - Finite horizon: $h \in \mathbb{N}$
    - infinite horizon: $h = \infty$
- Goal: find optimal policy π^* such that

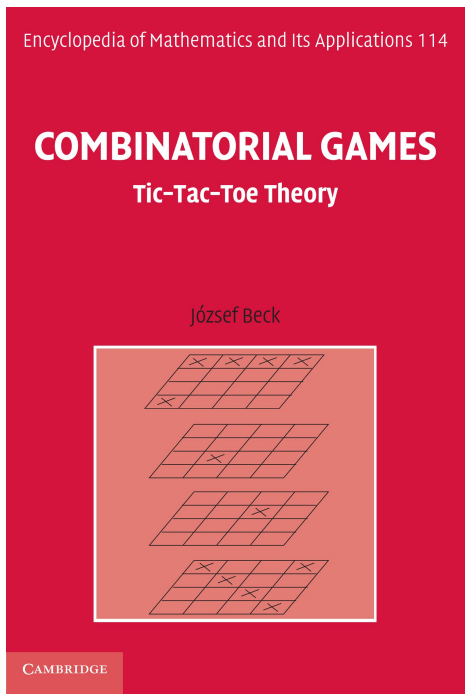
$$\pi^* = \operatorname*{argmax}_{\pi} \sum_{t=0}^{h} \gamma^t \mathbb{E}_{\pi} [r_t]$$

Policy Optimization

- **Markov Decision Process:**
  - Find optimal policy given transition and reward model
  - Execute policy found
- **Reinforcement learning:**
  - Learn an optimal policy while interacting with the environment

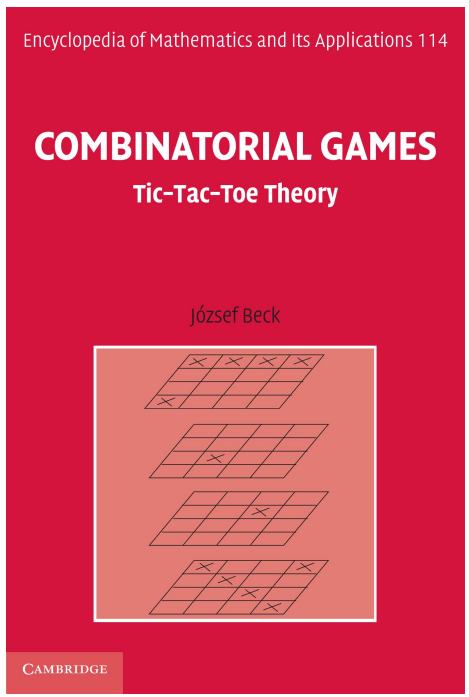
Policy Optimization

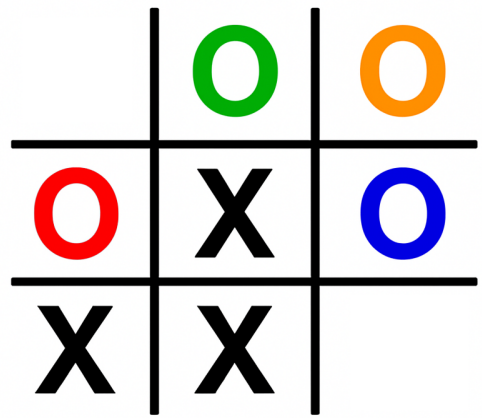
- **Markov Decision Process:**
  - Find optimal policy given transition and reward model
  - Execute policy found
- **Reinforcement learning:**
  - Learn an optimal policy while interacting with the environment



Policy Optimization

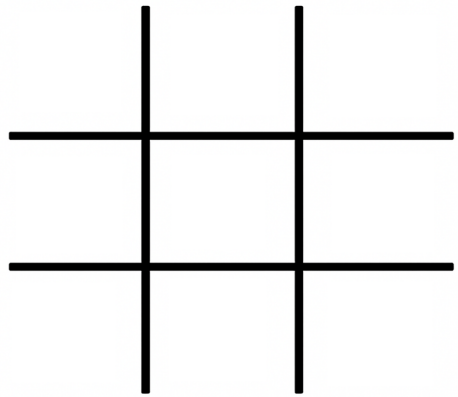
- **Markov Decision Process:**
  - Find optimal policy given transition and reward model
  - Execute policy found
- **Reinforcement learning:**
  - Learn an optimal policy while interacting with the environment





Example: MENACE

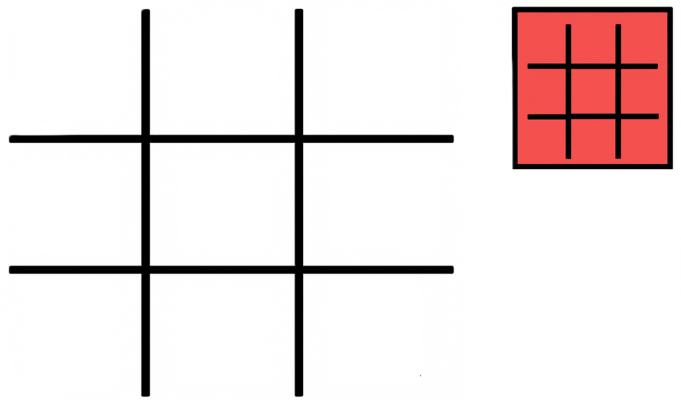
- Machine Educable Noughts And Crosses Engine (MENACE)
- Build by Donald Michie in 1961 using 304 matchboxes



Source: Matthew Scroggs

Example: MENACE

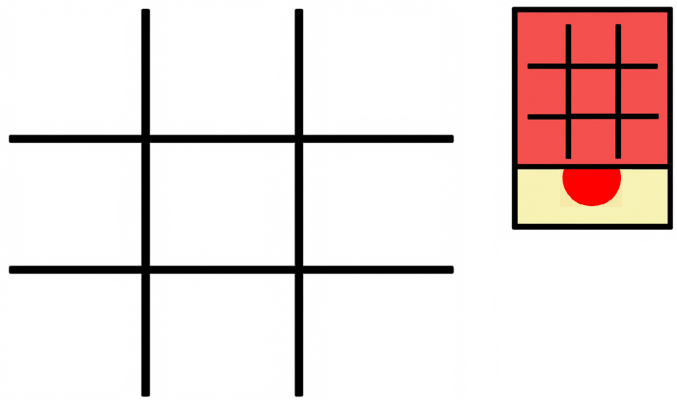
- Machine Educable Noughts And Crosses Engine (MENACE)
- Build by Donald Michie in 1961 using 304 matchboxes



Source: Matthew Scroggs

Example: MENACE

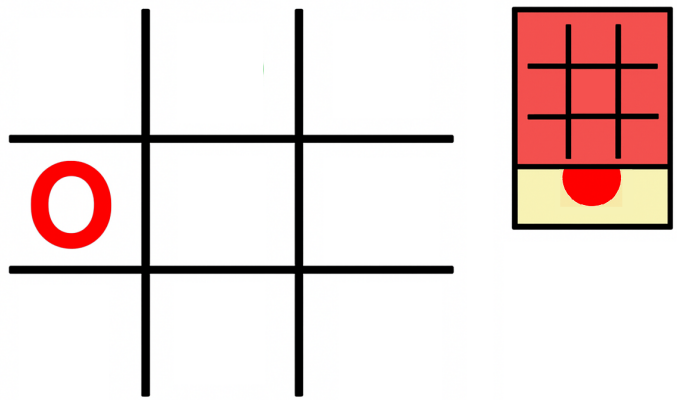
- Machine Educable Noughts And Crosses Engine (MENACE)
- Build by Donald Michie in 1961 using 304 matchboxes



Source: Matthew Scroggs

Example: MENACE

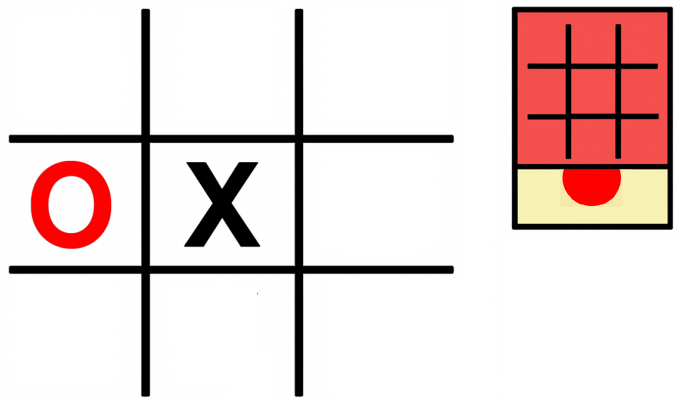
- Machine Educable Noughts And Crosses Engine (MENACE)
- Build by Donald Michie in 1961 using 304 matchboxes



Source: Matthew Scroggs

Example: MENACE

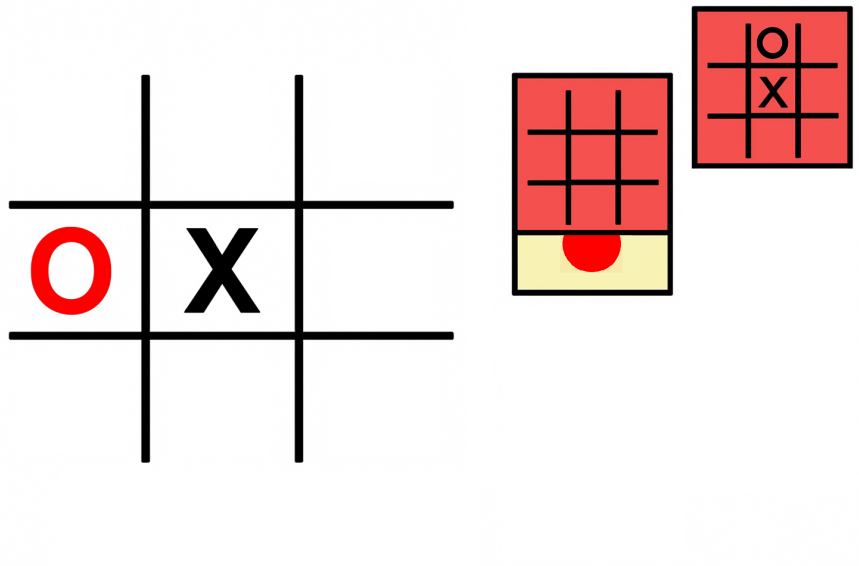
- Machine Educable Noughts And Crosses Engine (MENACE)
- Build by Donald Michie in 1961 using 304 matchboxes



Source: Matthew Scroggs

Example: MENACE

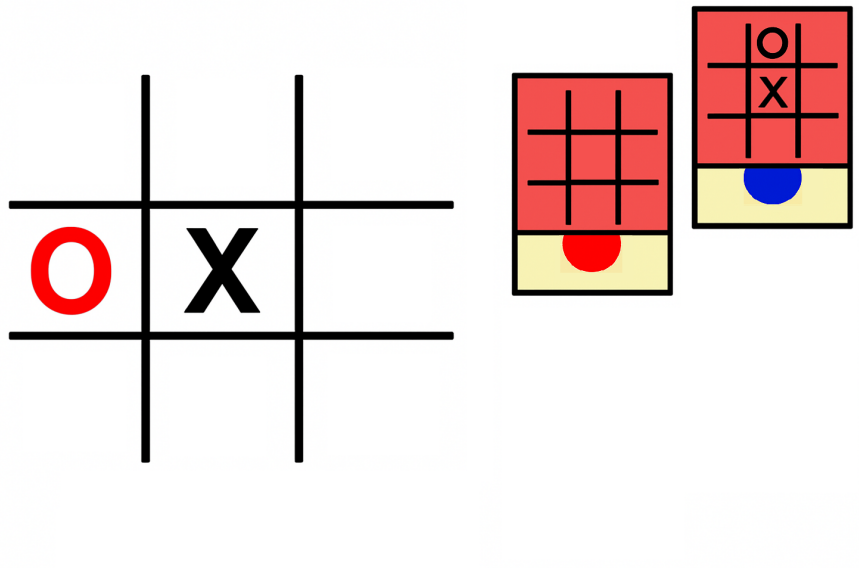
- Machine Educable Noughts And Crosses Engine (MENACE)
- Build by Donald Michie in 1961 using 304 matchboxes



Source: Matthew Scroggs

Example: MENACE

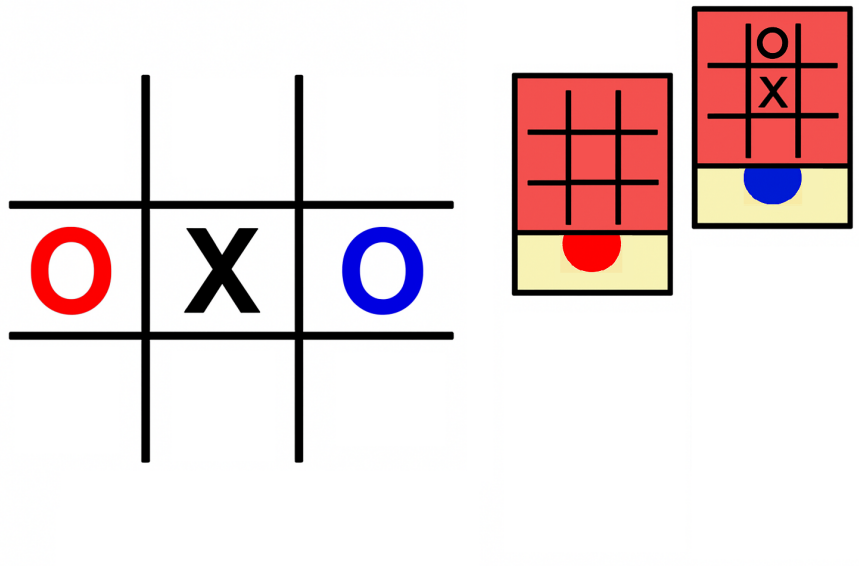
- Machine Educable Noughts And Crosses Engine (MENACE)
- Build by Donald Michie in 1961 using 304 matchboxes



Source: Matthew Scroggs

Example: MENACE

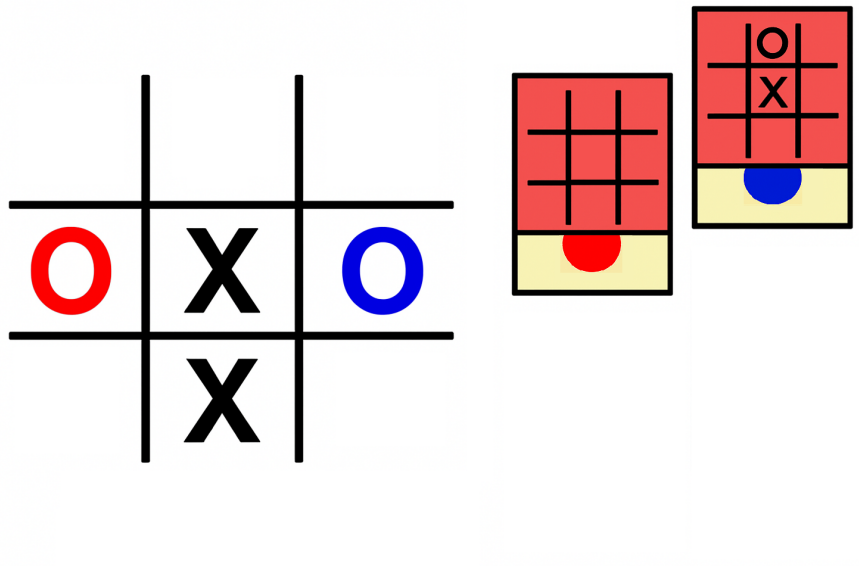
- Machine Educable Noughts And Crosses Engine (MENACE)
- Build by Donald Michie in 1961 using 304 matchboxes



Source: Matthew Scroggs

Example: MENACE

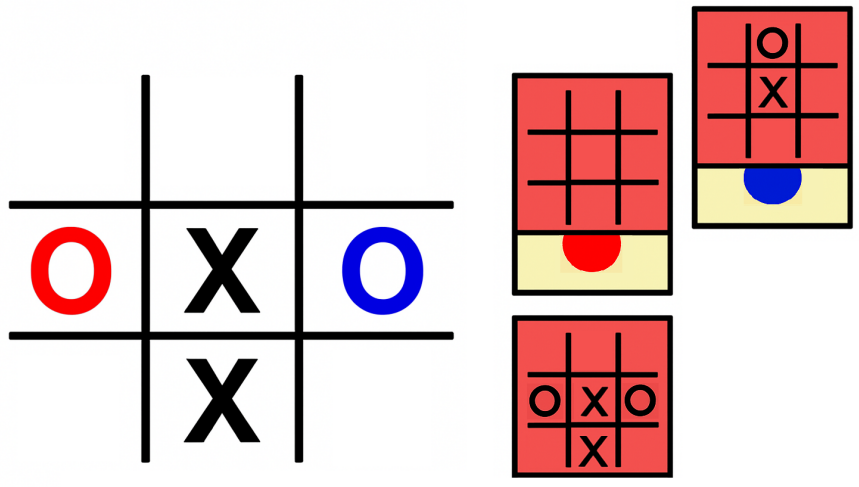
- Machine Educable Noughts And Crosses Engine (MENACE)
- Build by Donald Michie in 1961 using 304 matchboxes



Source: Matthew Scroggs

Example: MENACE

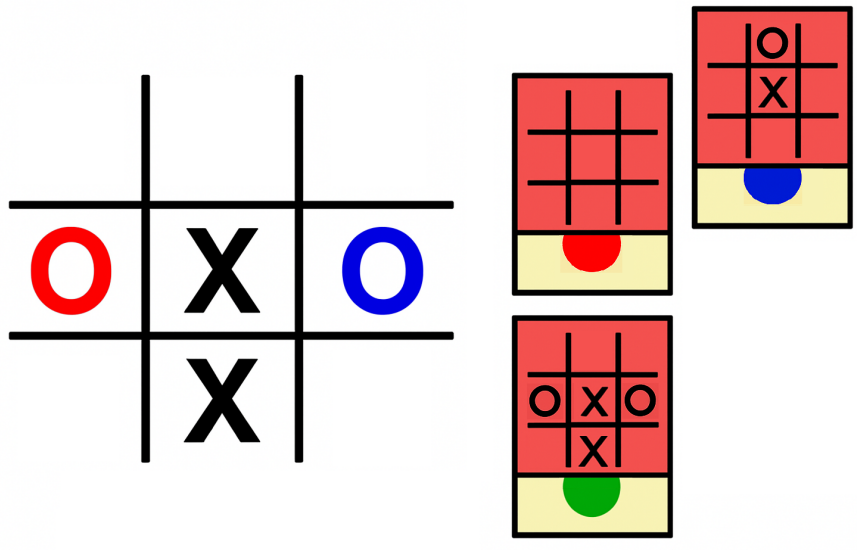
- Machine Educable Noughts And Crosses Engine (MENACE)
- Build by Donald Michie in 1961 using 304 matchboxes



Source: Matthew Scroggs

Example: MENACE

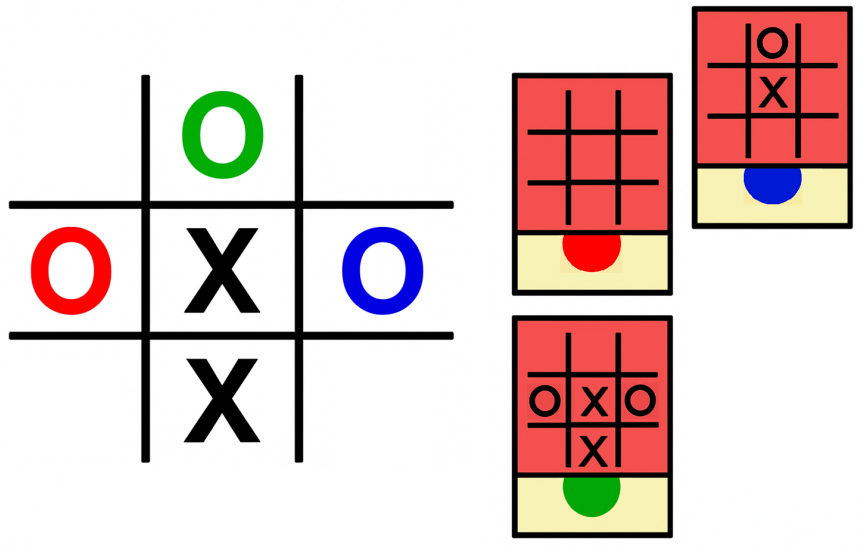
- Machine Educable Noughts And Crosses Engine (MENACE)
- Build by Donald Michie in 1961 using 304 matchboxes



Source: Matthew Scroggs

Example: MENACE

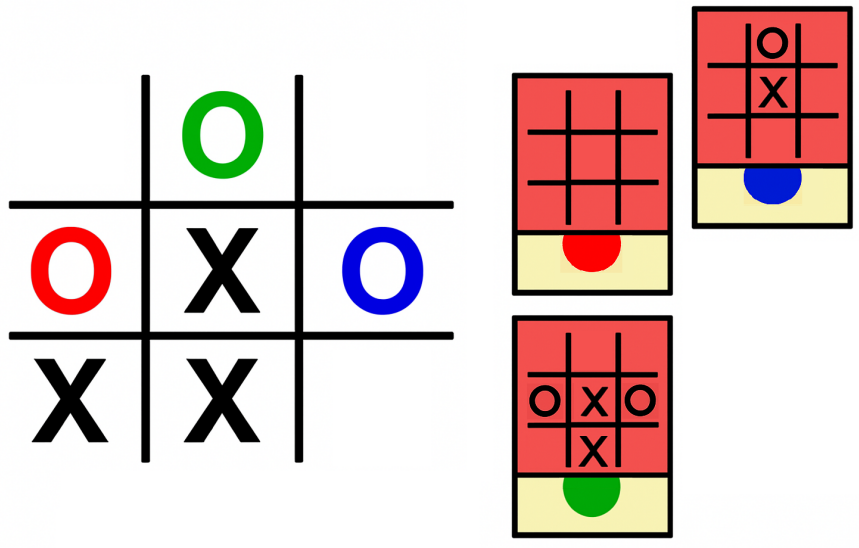
- Machine Educable Noughts And Crosses Engine (MENACE)
- Build by Donald Michie in 1961 using 304 matchboxes



Source: Matthew Scroggs

Example: MENACE

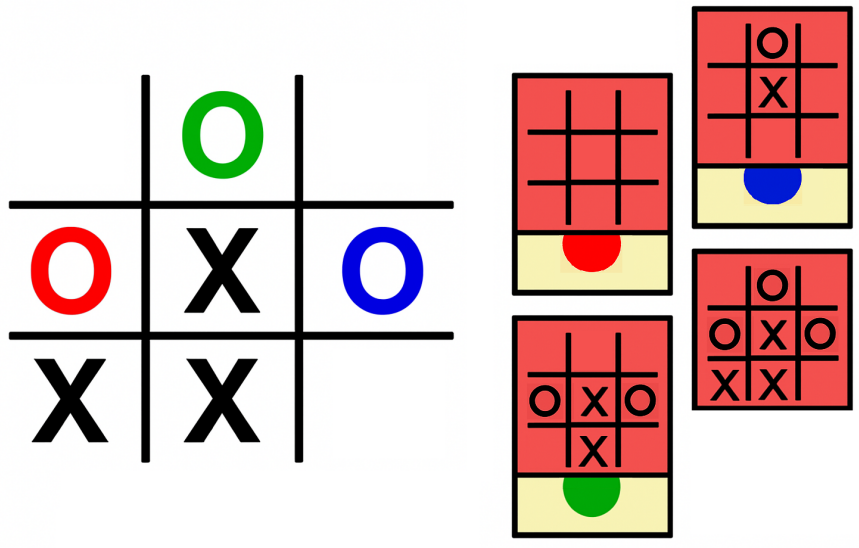
- Machine Educable Noughts And Crosses Engine (MENACE)
- Build by Donald Michie in 1961 using 304 matchboxes



Source: Matthew Scroggs

Example: MENACE

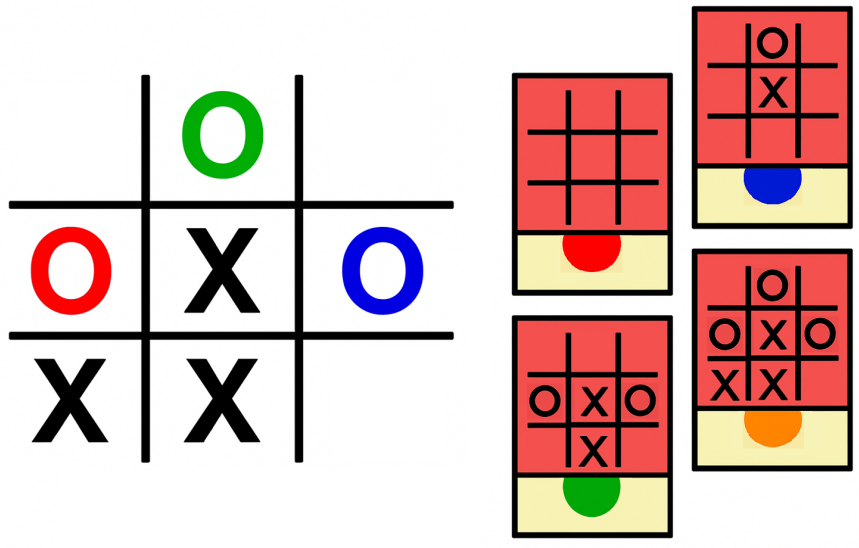
- Machine Educable Noughts And Crosses Engine (MENACE)
- Build by Donald Michie in 1961 using 304 matchboxes



Source: Matthew Scroggs

Example: MENACE

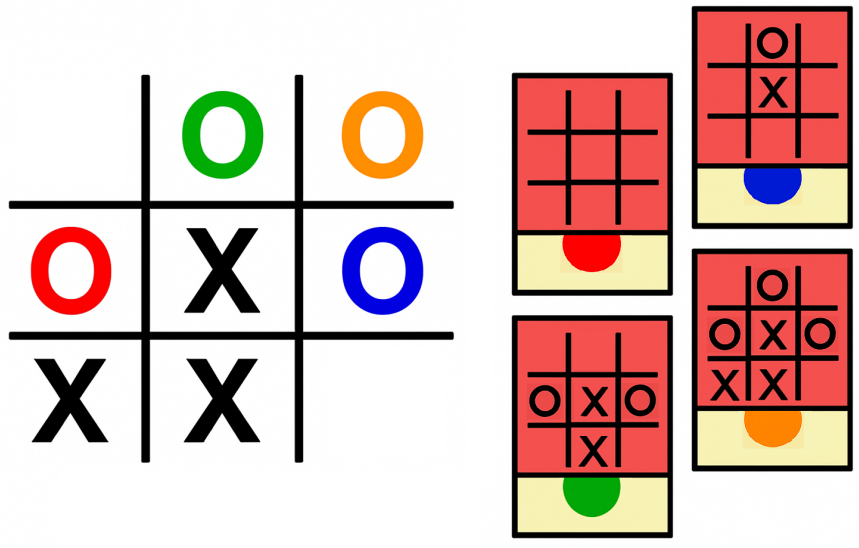
- Machine Educable Noughts And Crosses Engine (MENACE)
- Build by Donald Michie in 1961 using 304 matchboxes



Source: Matthew Scroggs

Example: MENACE

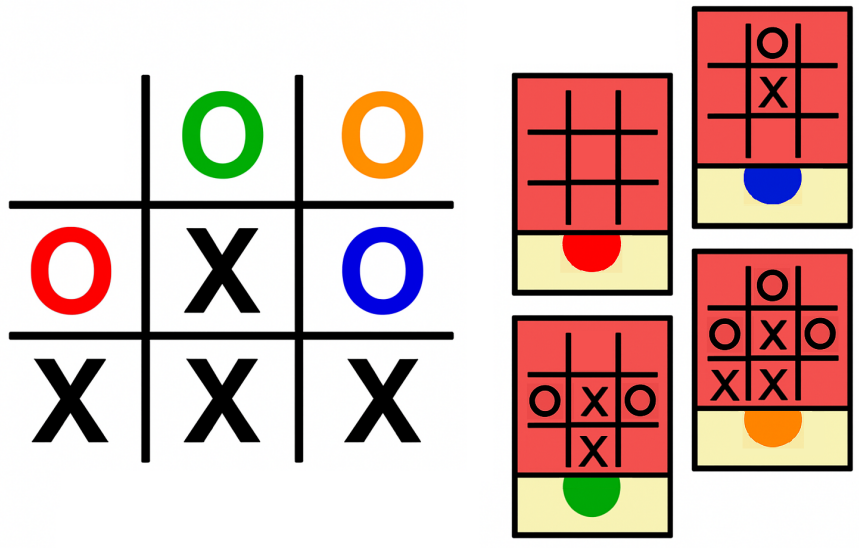
- Machine Educable Noughts And Crosses Engine (MENACE)
- Build by Donald Michie in 1961 using 304 matchboxes



Source: Matthew Scroggs

Example: MENACE

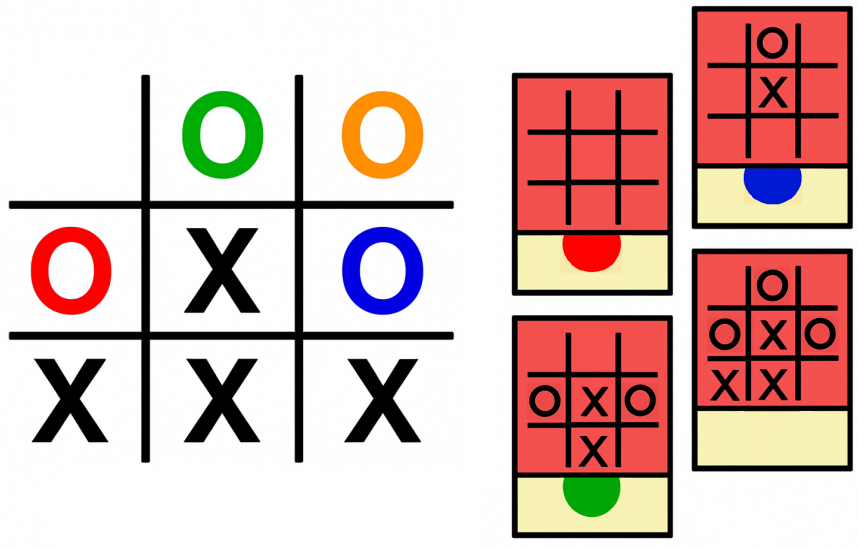
- Machine Educable Noughts And Crosses Engine (MENACE)
- Build by Donald Michie in 1961 using 304 matchboxes



Source: Matthew Scroggs

Example: MENACE

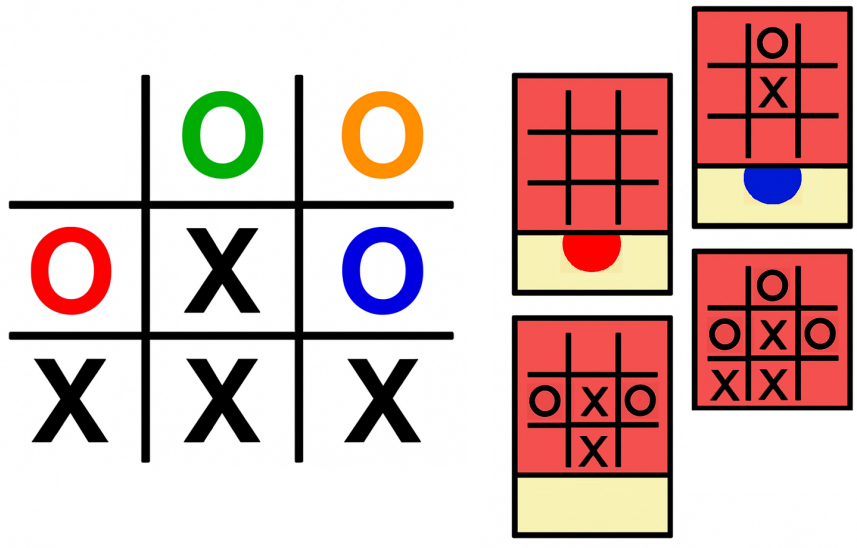
- Machine Educable Noughts And Crosses Engine (MENACE)
- Build by Donald Michie in 1961 using 304 matchboxes



Source: Matthew Scroggs

Example: MENACE

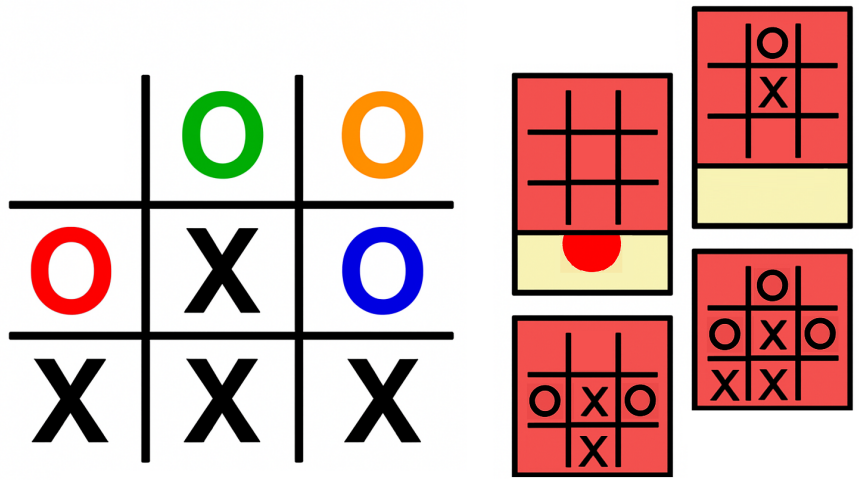
- Machine Educable Noughts And Crosses Engine (MENACE)
- Build by Donald Michie in 1961 using 304 matchboxes



Source: Matthew Scroggs

Example: MENACE

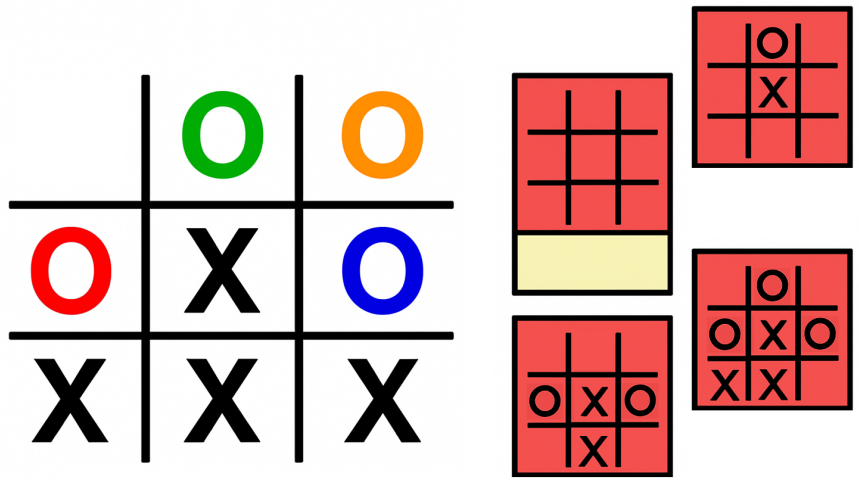
- Machine Educable Noughts And Crosses Engine (MENACE)
- Build by Donald Michie in 1961 using 304 matchboxes



Source: Matthew Scroggs

Example: MENACE

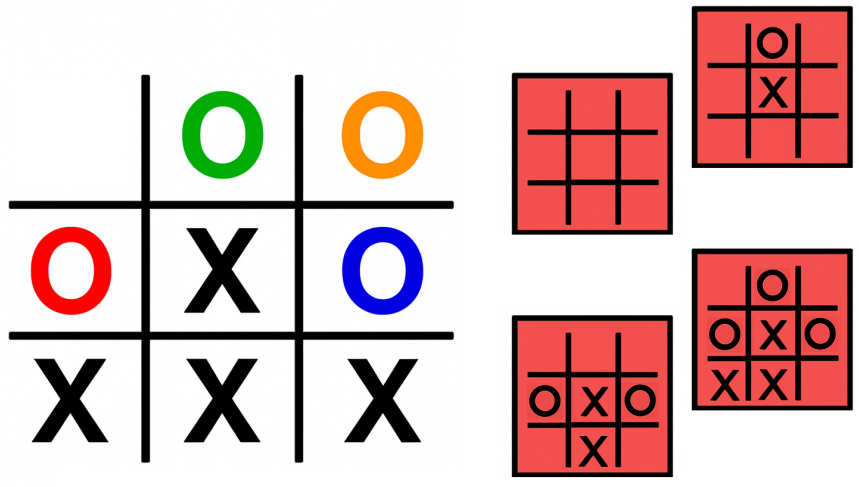
- Machine Educable Noughts And Crosses Engine (MENACE)
- Build by Donald Michie in 1961 using 304 matchboxes



Source: Matthew Scroggs

Example: MENACE

- Machine Educable Noughts And Crosses Engine (MENACE)
- Build by Donald Michie in 1961 using 304 matchboxes



Source: Matthew Scroggs

Lose	$\rightarrow$	Remove the bead from each box
Win	$\rightarrow$	Add three beads to each box
Draw	$\rightarrow$	Add one bead to each box

Important Components in Reinforcement Learning

Reinforcement learning agents may or may not include the following components:

- **Model:** $\mathbb{P}(s' \mid s, a), \mathbb{P}(r \mid s, a)$
  - Environment dynamics and rewards
- **Policy:** $\pi(s)$
  - Agent action choices
- **Value function:** $V(s)$
  - Expected total rewards of the agent's policy

Important Components in Reinforcement Learning

Reinforcement learning agents may or may not include the following components:

- **Model:** $\mathbb{P}(s' \mid s, a), \mathbb{P}(r \mid s, a)$
  - Environment dynamics and rewards
- **Policy:** $\pi(s)$
  - Agent action choices
- **Value function:** $V(s)$
  - Expected total rewards of the agent's policy
- **Quality function:** $Q(s, a)$
  - Expected total rewards of taking a specific action in a given state

Bellman's Equation

- Optimal state value function $V^*(s)$

$$V^*(s) = \max_a E[r \mid s, a] + \gamma \sum_{s'} \mathbb{P}\left(s' \mid s, a\right) V^*\left(s'\right)$$

- Optimal state-action value function $Q^*(s, a)$

$$Q^*(s, a) = E[r \mid s, a] + \gamma \sum_{s'} \mathbb{P}\left(s' \mid s, a\right) \max_{a'} Q^*\left(s', a'\right)$$

where $V^*(s) = \max_a Q^*(s, a)$

$$\pi^*(s) = \operatorname*{argmax}_a Q^*(s, a)$$

Categorizing RL Agents

Categorizing RL Agents

Value based

- No policy (implicit)
- Value function

Policy based

- Policy
- No value function

Actor critic

- Policy
- Value function

Model based

- Transition and reward model

Model free

- No transition model (implicit)
- No reward model (implicit)

Categorizing RL Agents

Value based

- No policy (implicit)
- Value function

Policy based

- Policy
- No value function

Actor critic

- Policy
- Value function

Model based

- Transition and reward model

Model free

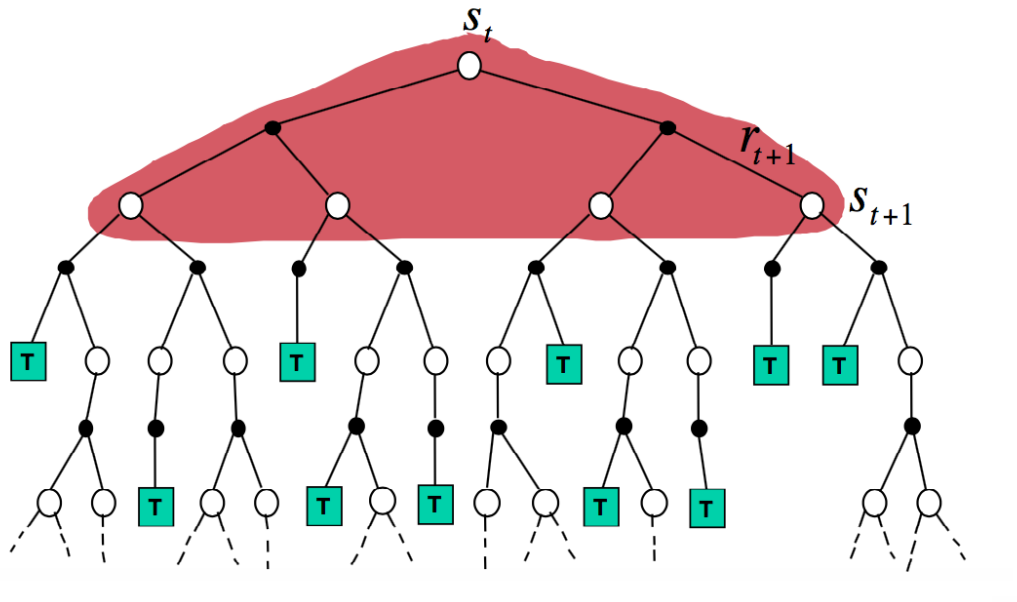
- No transition model (implicit)
- No reward model (implicit)

Imitation Learning

- No transition model (implicit)
- Reward model implicit through experts

Dynamic Programming Backup

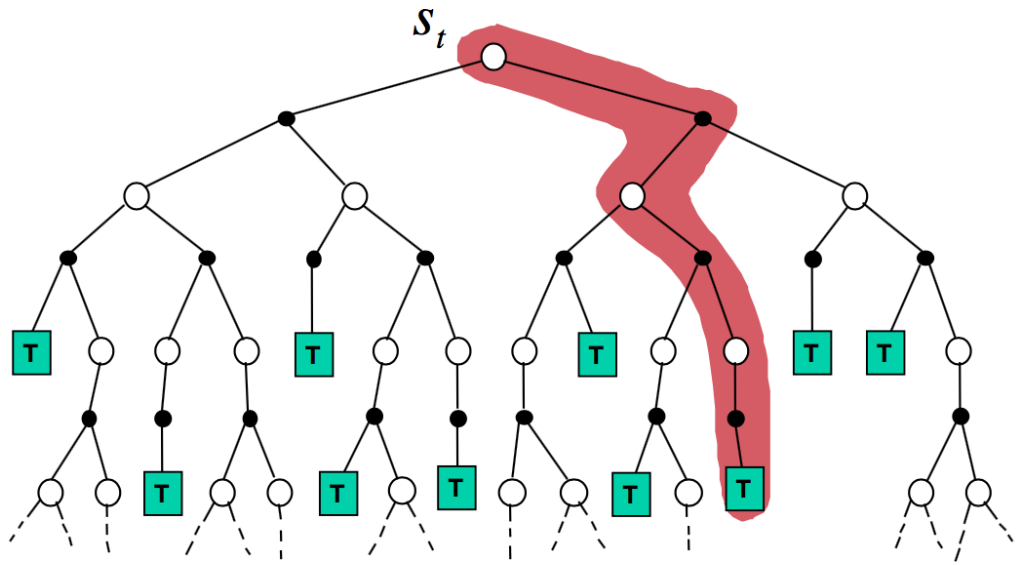
$$V(S_t) \leftarrow \mathbb{E}_\pi \left[R_{t+1} + \gamma V(S_{t+1}) \right]$$



Source: David Silver

Monte Carlo Backup

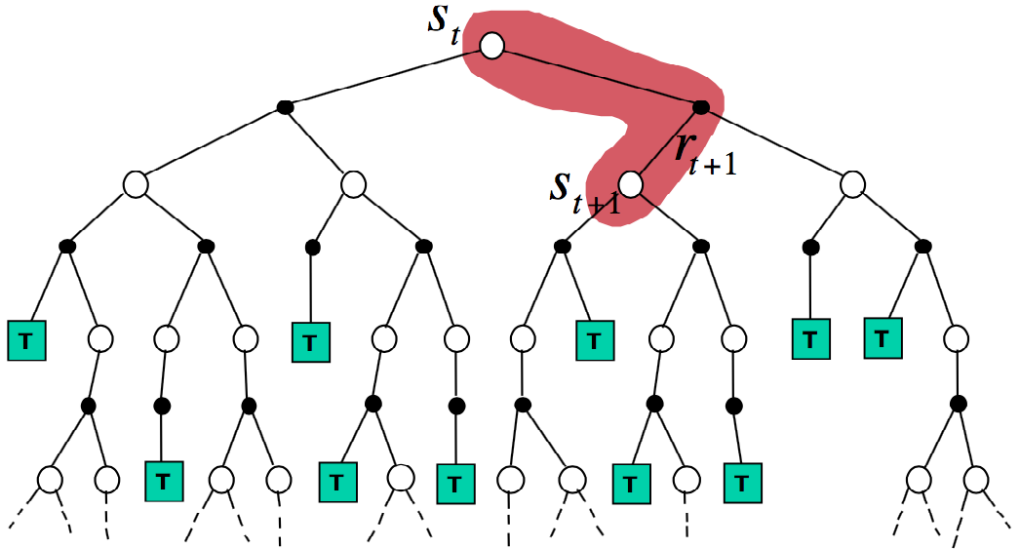
$$V(S_t) \leftarrow V(S_t) + \alpha\,(G_t - V(S_t))$$



Source: David Silver

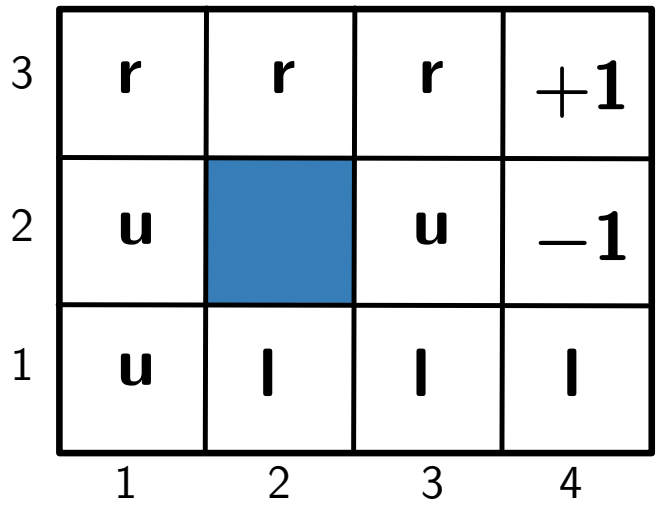
Temporal Difference Backup

$$V(S_t) \leftarrow V(S_t) + \alpha \left(R_{t+1} + \gamma V(S_{t+1}) - V(S_t)\right)$$



Source: David Silver

Toy Maze Example



Start state: (1,1)
Terminal states: (4,2), (4,3)
No discount: γ=1

Reward is -0.04 for non-terminal states

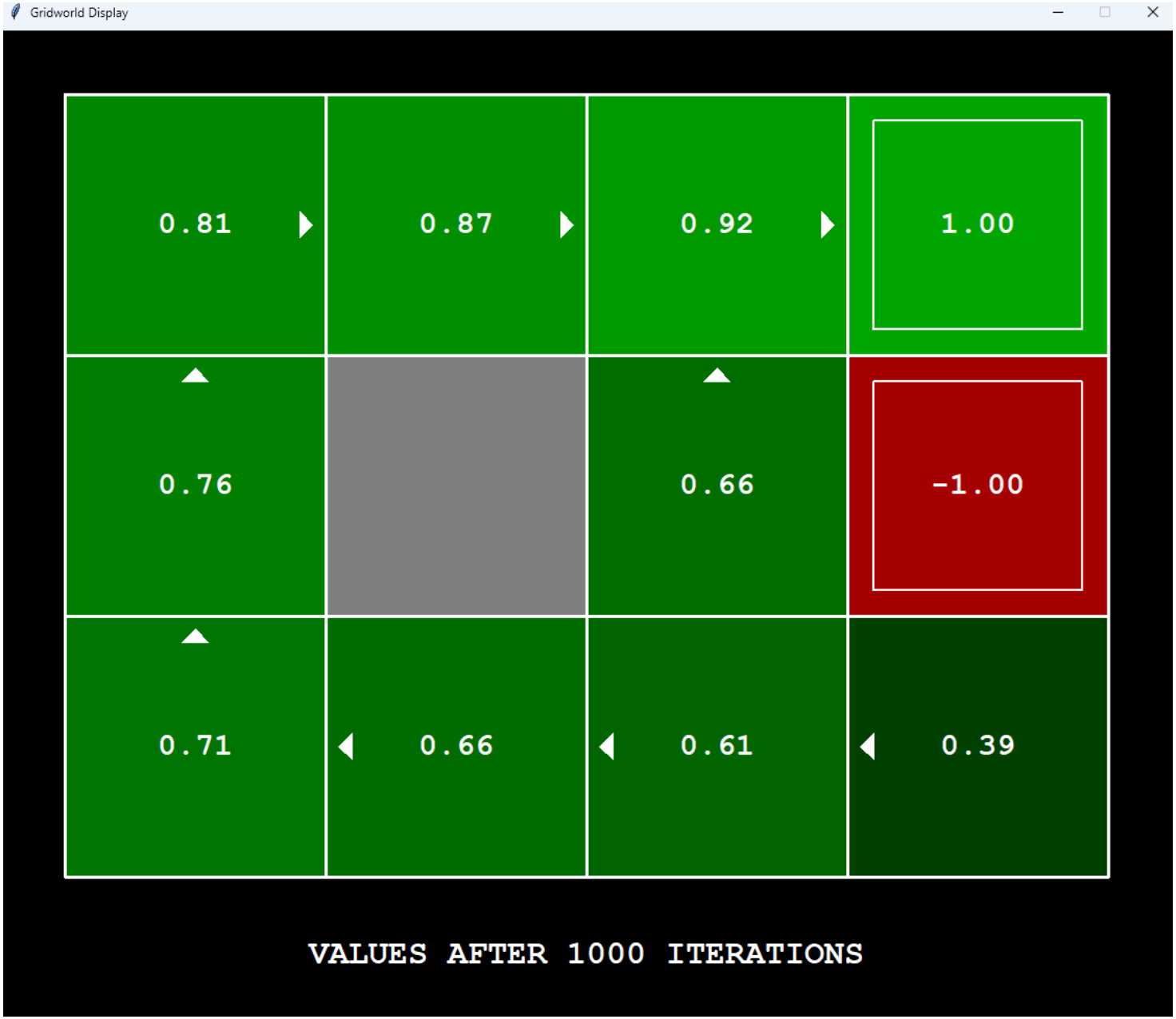
Four actions:

- up (**u**),
- left (**l**),
- right (**r**),
- down (**d**)

Do not know the transition probabilities

What is the value $V(s)$ of being in state s

Toy Maze Example (No Learning, Noise 20%)



References I

DeNero, J., D. Klein, B. Miller, N. Hay, and P. Abbeel (2013): "The Pacman AI Projects," http://inst.eecs.berkeley.edu/~cs188/pacman/, Developed at UC Berkeley. Core by John DeNero and Dan Klein; student autograding by Brad Miller, Nick Hay, and Pieter Abbeel.

Poupart, P. (2025): "Pascal Poupart's Homepage," https://cs.uwaterloo.ca/~ppoupart/, Accessed: 2025-05-24.

Russell, S. J., and P. Norvig (2016): *Artificial intelligence: a modern approach*. Pearson.

Sigaud, O., and O. Buffet (2013): *Markov decision processes in artificial intelligence*. John Wiley & Sons.

Sutton, R. S., and A. G. Barto (2018): "Reinforcement learning: An introduction," *A Bradford Book*, Available at http://incompleteideas.net/book/the-book-2nd.html.

Szepesvári, C. (2022): *Algorithms for reinforcement learning*. Springer nature, Available at https://sites.ualberta.ca/~szepesva/RLBook.html.

Takeaways

Takeaways

- "By three methods we may learn wisdom:
  - First, by reflection, which is ~~noblest~~ model-based RL;
  - Second, by imitation, which is ~~easiest~~ imitation learning; and
  - third by experience, which is the ~~bitterest~~ model free RL."

- RL agent types:
  - value-based,
  - policy-based,
  - value-policy-based (actor-critic),
  - model-based,
  - model-free
  - imitation learning