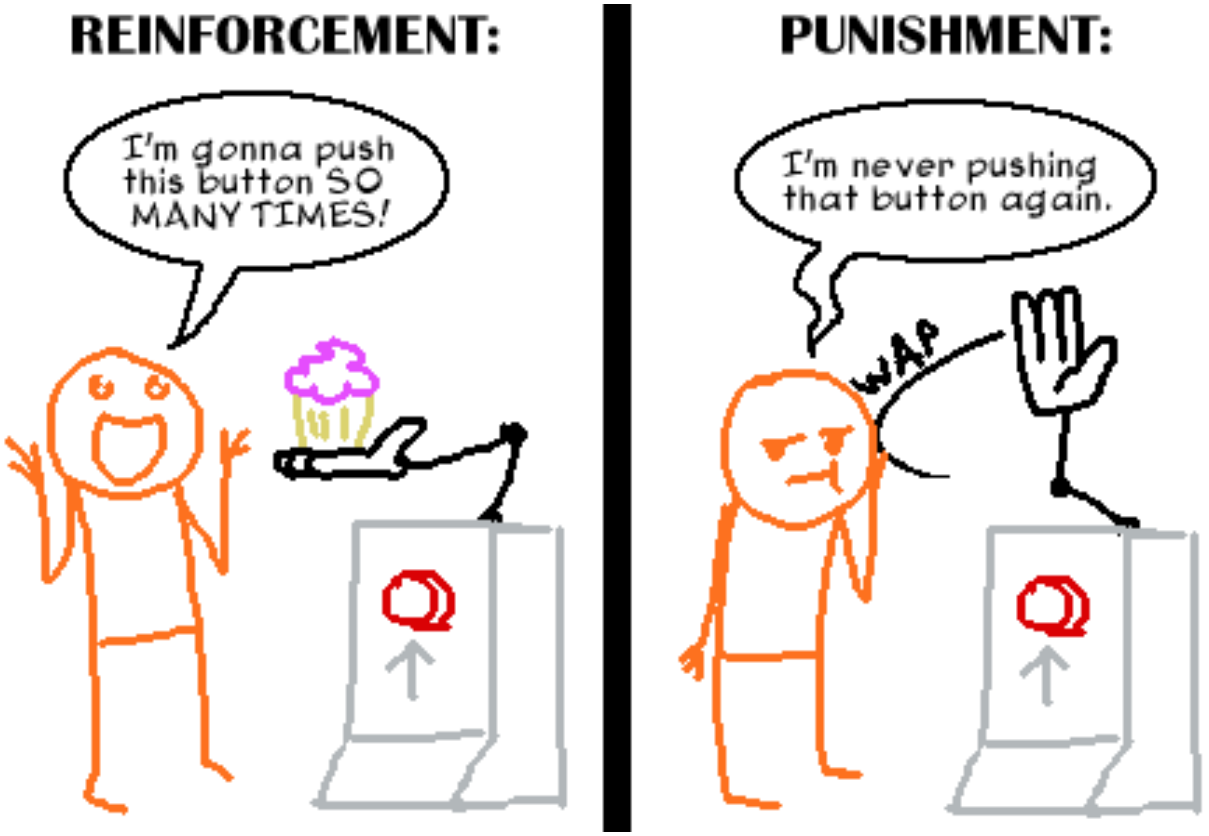
Reinforcement Learning

27 Jun 2022

John Tan Chong Min

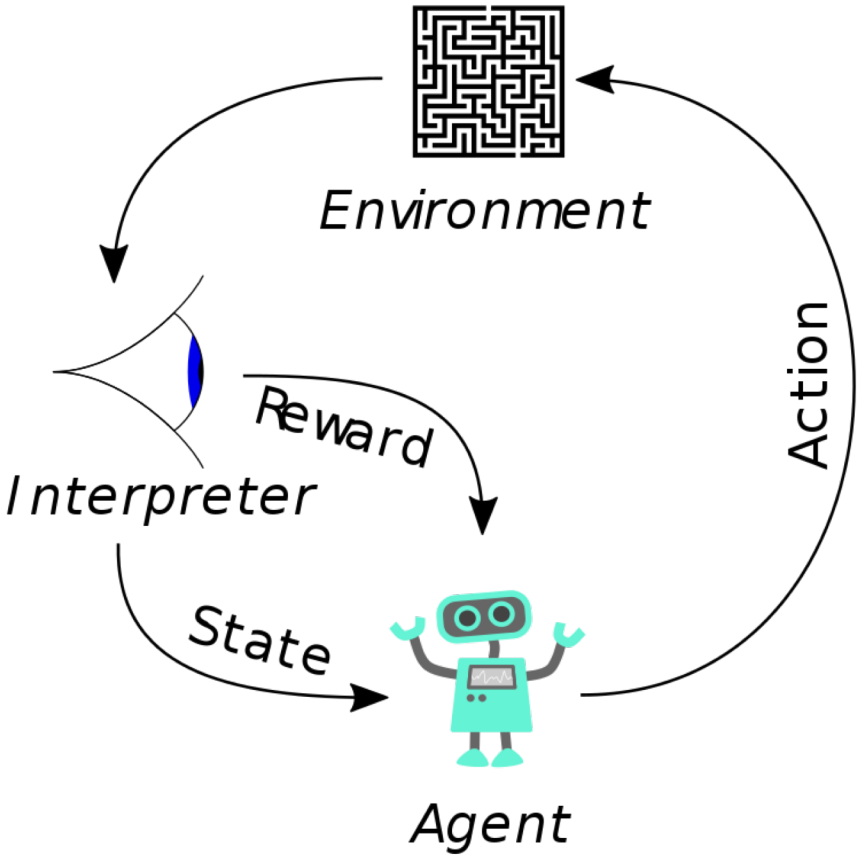
Reinforcement Learning in **one slide**



Taken from: https://kazerad.tumblr.com/post/99022123468/shepherd-of-the-masked

Motivation

- We learn from experience of our actions
  - Key thing missing from supervised/unsupervised learning
- For a Reinforcement Learning (RL) agent, we have
  - State: *s*
  - Action: *a*
  - Reward: *r*

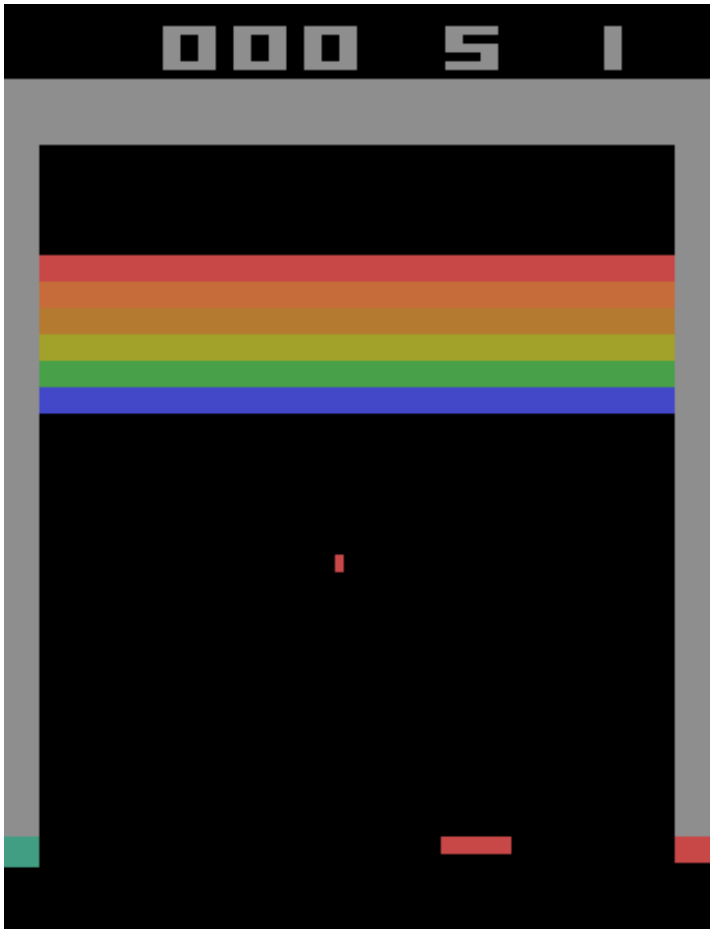


Let’s see RL in action!

Video on Atari AI

- Takes in raw pixels as input
- Outputs an action!

- https://www.youtube.com/watch?v=TmPfTpjtdgg



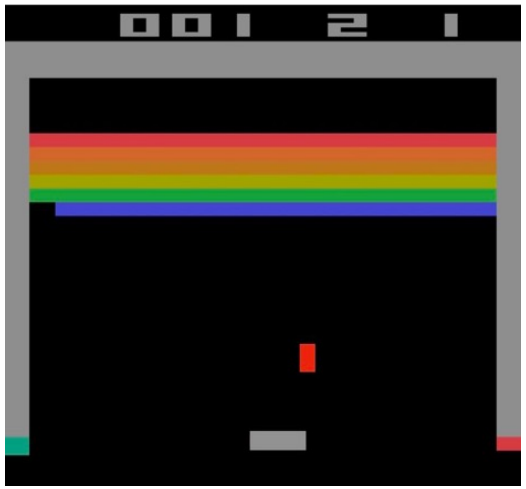
Agent Running/Walking/Jumping

- Sensors as inputs
- Outputs the joint configuration

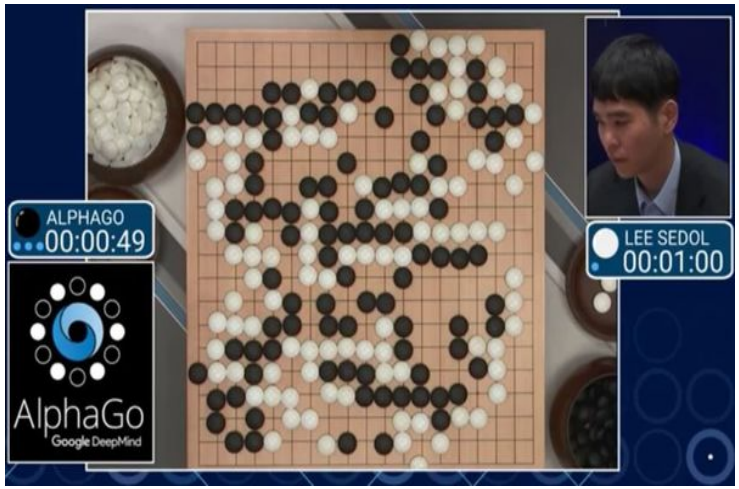
- https://www.youtube.com/watch?v=gn4nRCC9TwQ

Breakthroughs in Game AI

Chess:
Minimax Search + Alpha-Beta Pruning



Atari 2600 Games:
Deep Q-Network (DQN)



Go:
Monte Carlo Tree Search (MCTS)
+ Neural Networks (NN)

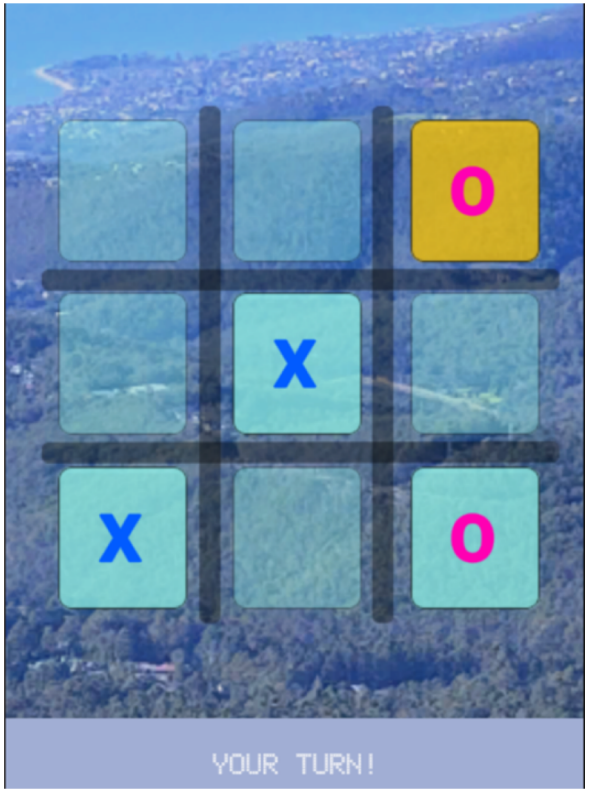
Increasing abstraction

Decreasing expert knowledge from humans

Let’s practice!

Identify State, Action, Reward

Example Game 1: Tic Tac Toe

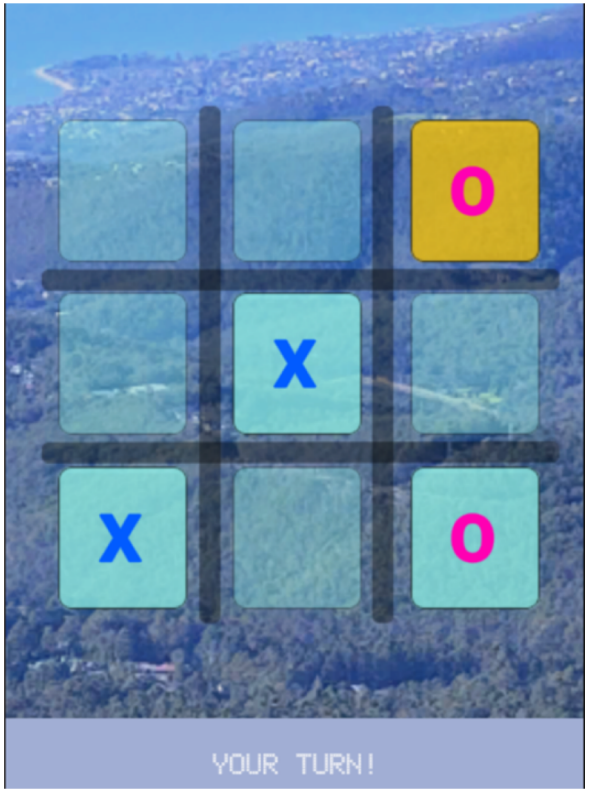


Tic Tac Toe: Win by connecting 3 in a row

Identify the

- **State**
- **Action**
- **Reward**

Example Game 1: Tic Tac Toe



Tic Tac Toe: Win by connecting 3 in a row

State:

Board position

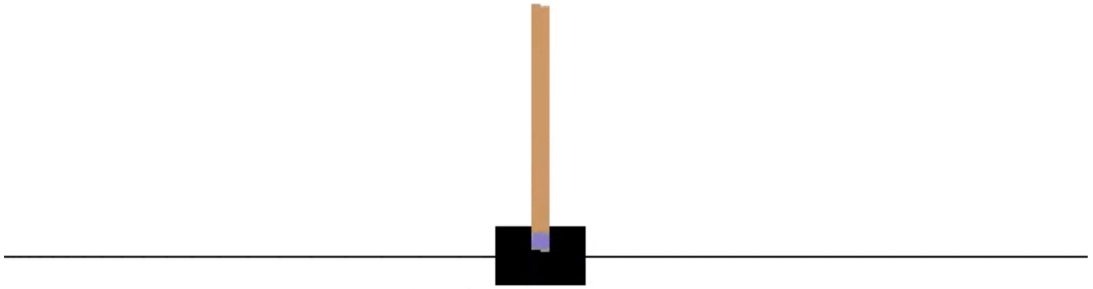
Action:

An empty square to put a piece in

Reward:

+1 for win, -1 for loss, 0 for draw

Example Game 2: Cart Pole

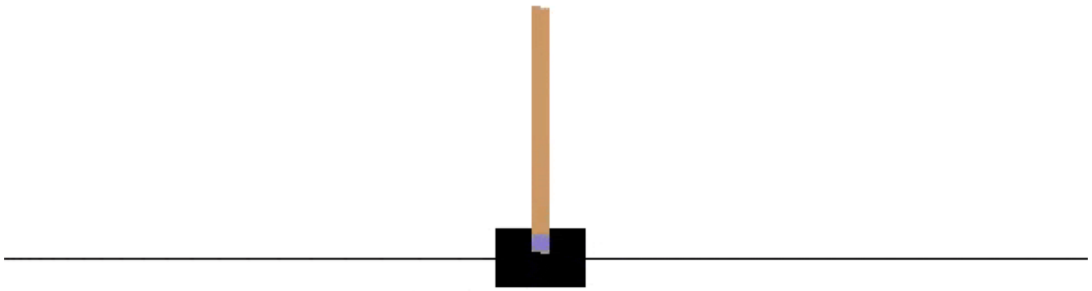


Cart Pole: Balance a pole on a cart for as long as possible

Identify the

- **State**
- **Action**
- **Reward**

Example Game 2: Cart Pole



Cart Pole: Balance a pole on a cart for as long as possible

State:
Cart position, velocity
Pole angle, velocity

Action:
Left or Right

Reward:
+1 for every timestep pole is balanced

Example Game 3: Atari Breakout

Atari Breakout: Destroy all bricks without losing the ball

Identify the

- **State**
- **Action**
- **Reward**

Example Game 3: Atari Breakout

Atari Breakout: Destroy all bricks without losing the ball

State:
Ball position & velocity, paddle position & velocity, brick position, ball velocity
(OR grid of pixels)

Action:
Left or Right

Reward:
+1 for each brick destroyed

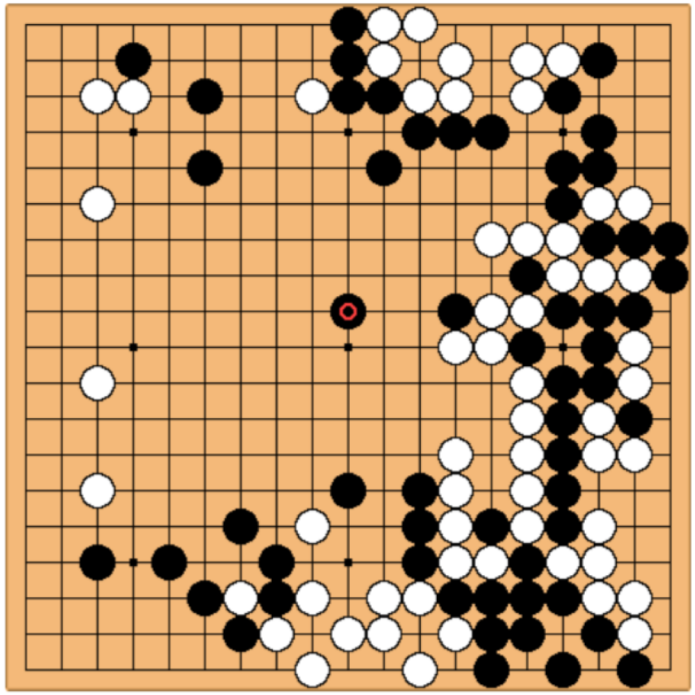
Types of Games

Deterministic vs Probabilistic

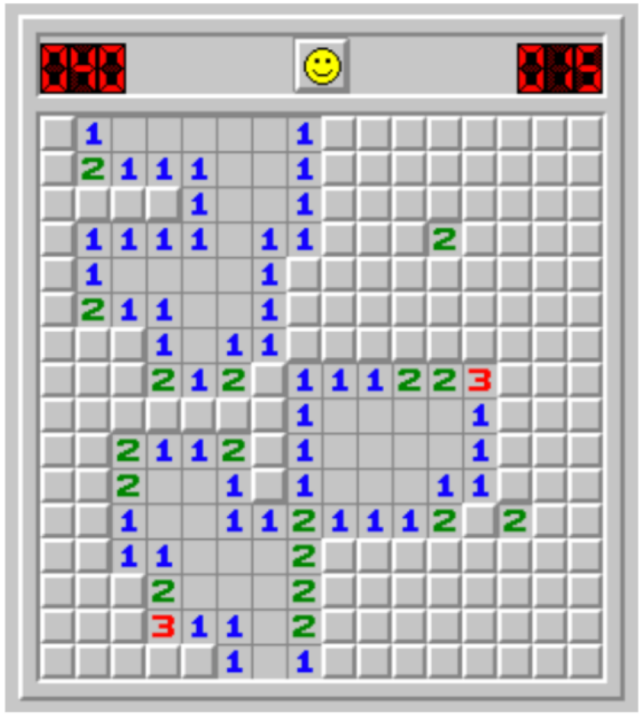
Discrete vs Continuous

Deterministic vs Probabilistic

Go: Deterministic

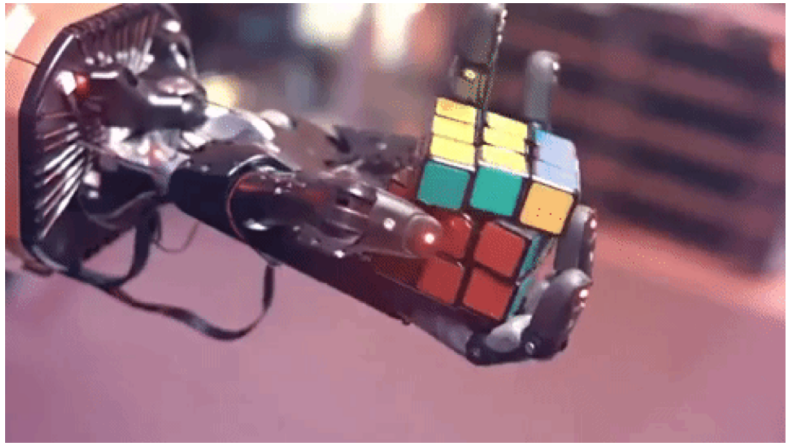


Minesweeper: Probabilistic

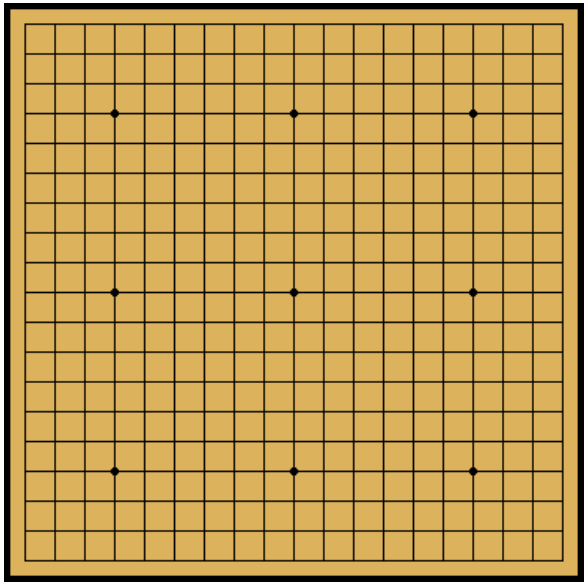


Discrete vs Continuous State/Action Space

Robotic Arm: Continuous



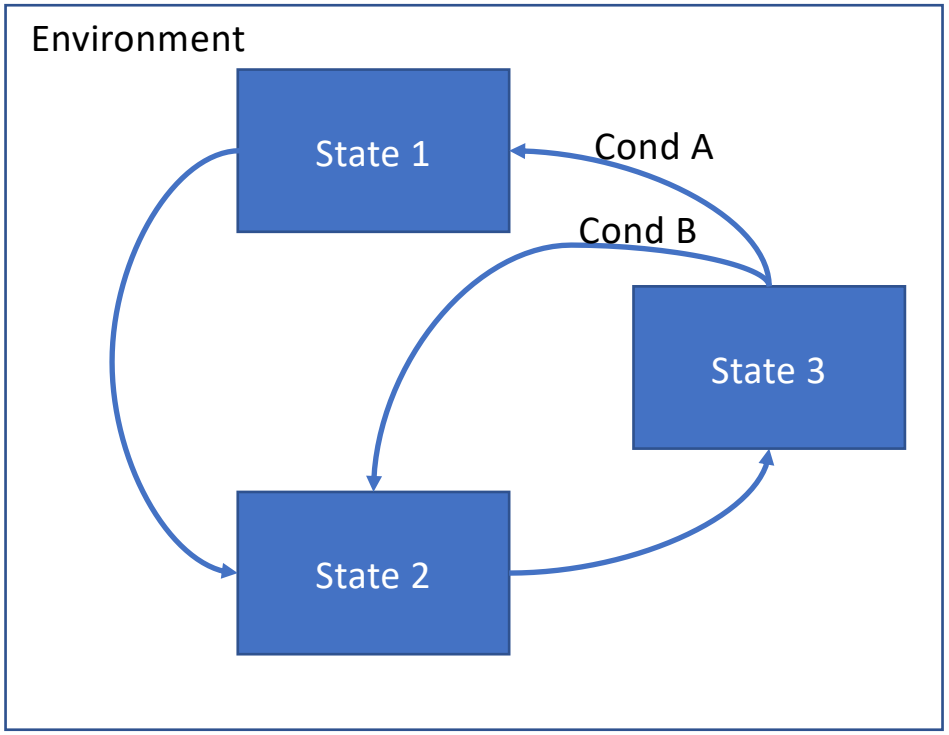
Go: Discrete



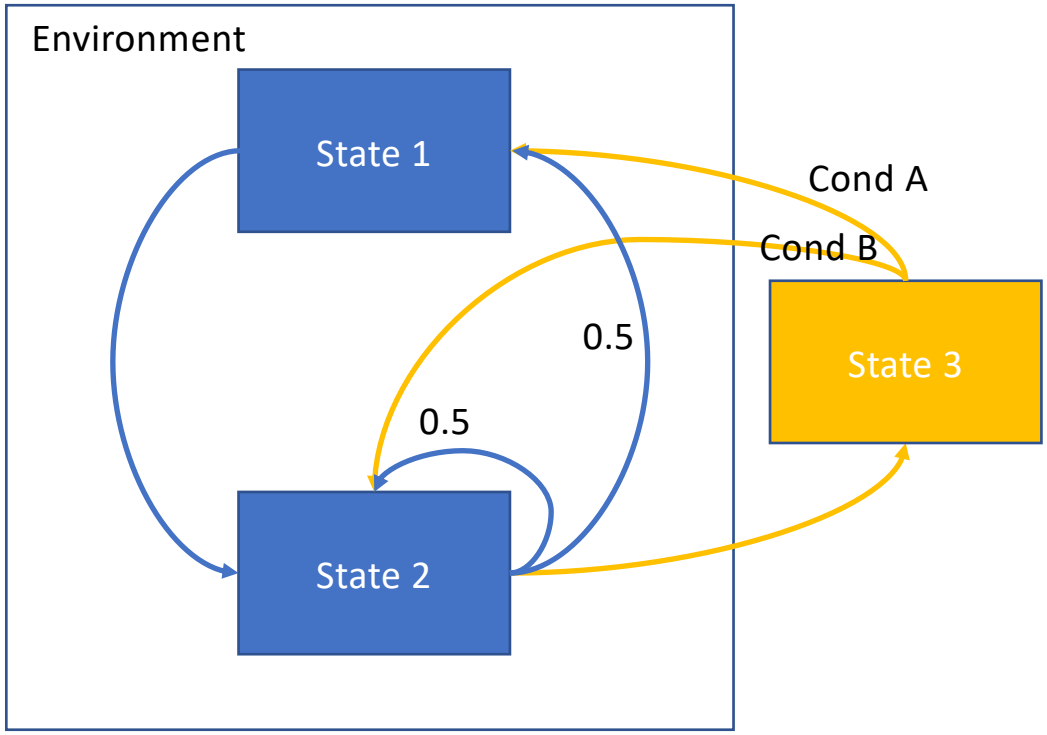
Markov Decision Process

A mathematical way to characterize games

State-Action Representation

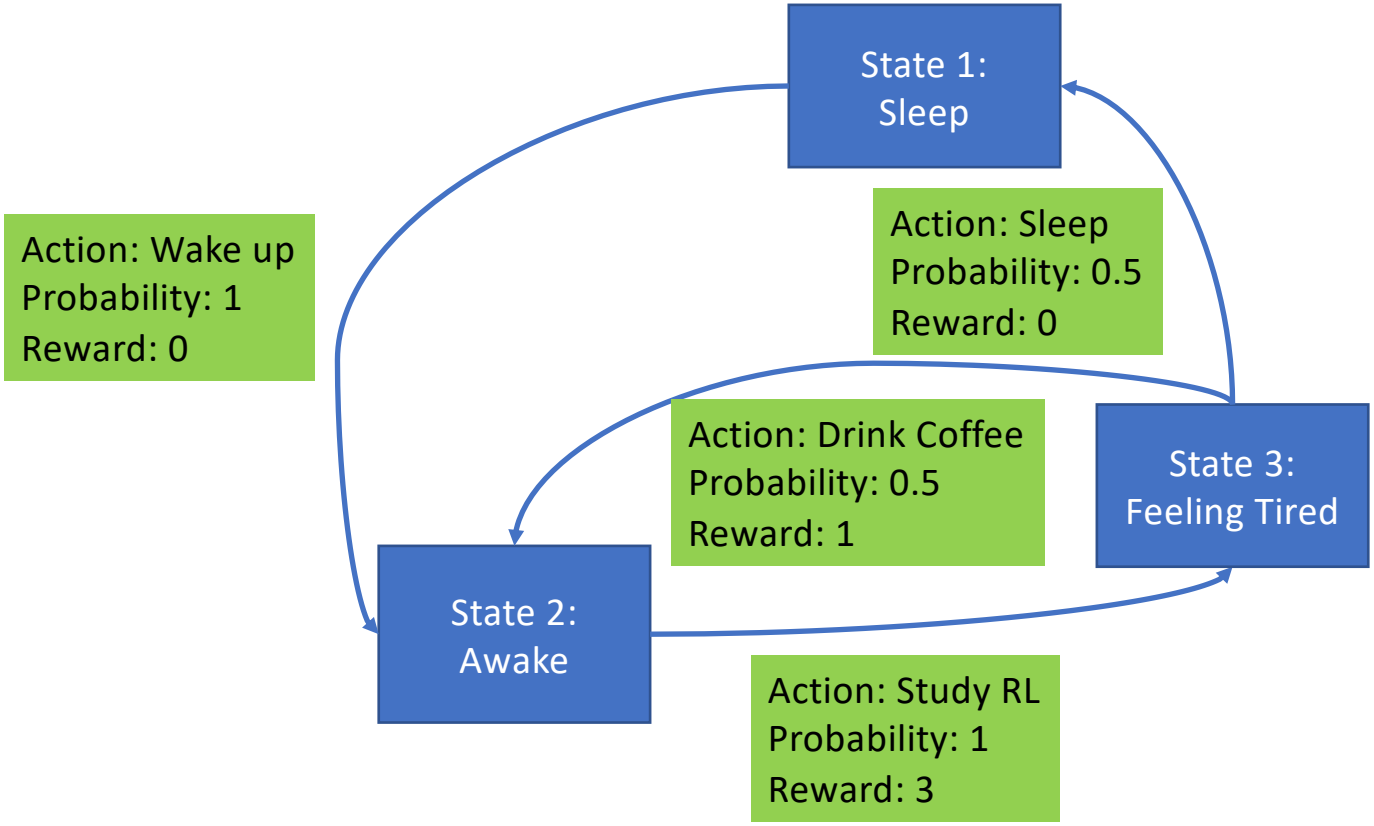


Deterministic



Probabilistic

Markov Decision Process (MDP)



MDP states are memory-free.

Knowing the state itself is sufficient to do planning, no need for past state histories.

Question: What if your game/environment requires a history of past states?

How to make decisions

Planning and Learning

How to make decisions?

- **Planning:** Sample possible outcomes
  - **Monte Carlo/Monte Carlo Tree Search (Week 1)**
  - Possible to combine with Neural Networks / Bootstrapping
    - AlphaGo / AlphaGo Zero

- **Learning:** Use future values to update earlier values (Bootstrapping)
  - Temporal Difference Learning (TD Learning)
  - Value/Policy Iteration
  - **Q-Learning / DQN (Week 2)**

How to make decisions?

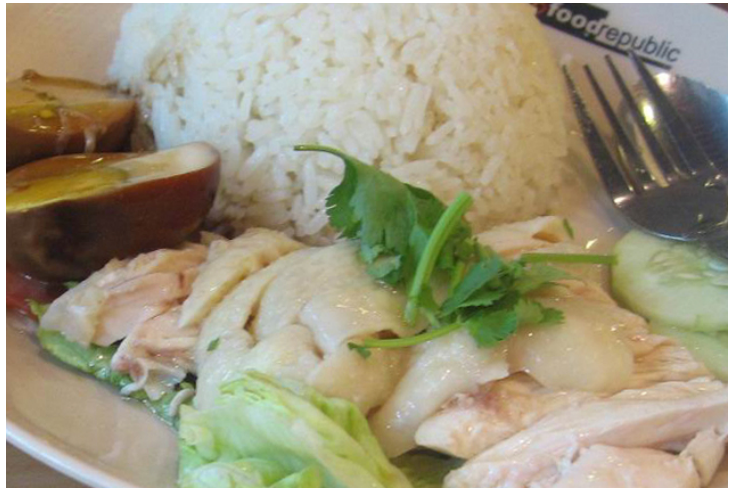
- Expert knowledge
  - Derive heuristics from human knowledge
  - Not RL, but good way to create a baseline competent agent

- Discrete vs Continuous Action Spaces
  - Discrete: MCTS / Q-Learning / DQN
  - Continuous: Policy-gradient based methods - PPO / A2C / A3C

Explore-Exploit

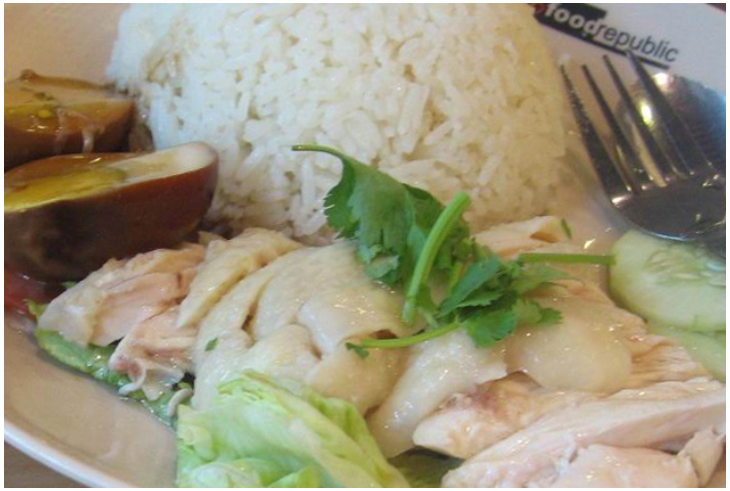
How to choose an optimal action?

How to decide?
Chicken rice vs Chicken Chop



VS

Explore-Exploit Tradeoff



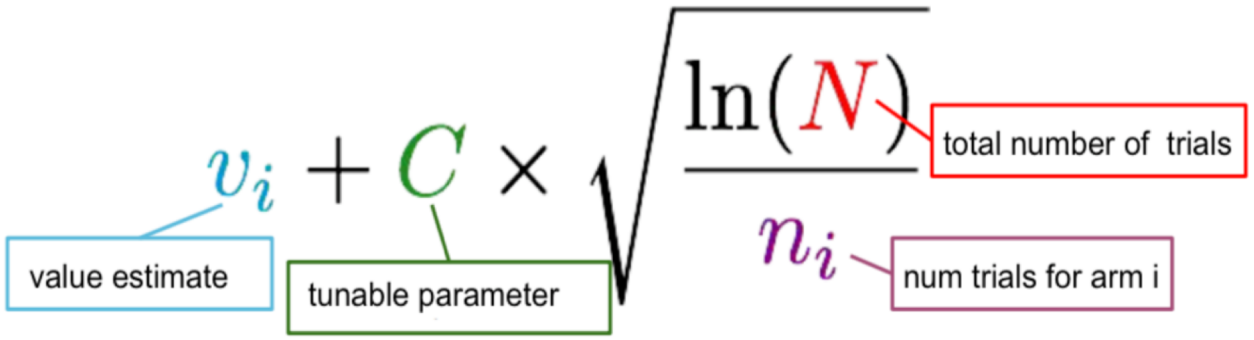
VS

We want to keep exploiting a good choice to reap its benefits,
But we must also ensure we explore other options sufficiently in case they are better.

Explore vs Exploit: Upper Confidence Bounds

UCB1 Formula (Auer et al 2002)

Choose arm *i* which maximises:

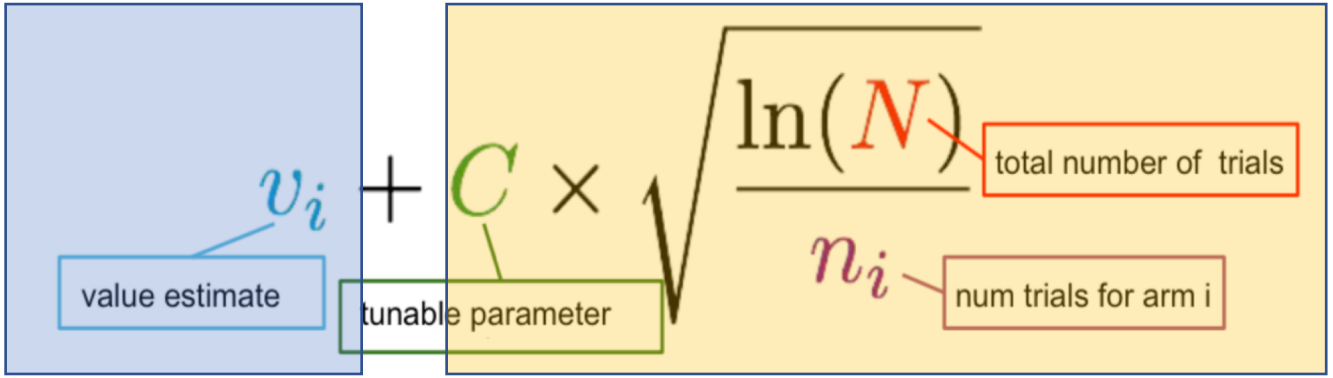


Proof: http://courses.cms.caltech.edu/cs101.2/slides/cs101.2-02-Bandits-notes.pdf

Explore vs Exploit: Upper Confidence Bounds

UCB1 Formula (Auer et al 2002)

Choose arm *i* which maximises:



Exploit

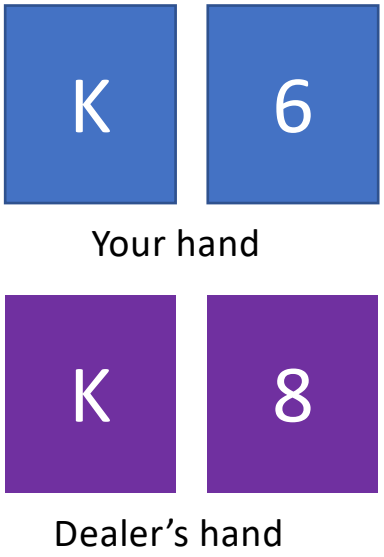
Explore

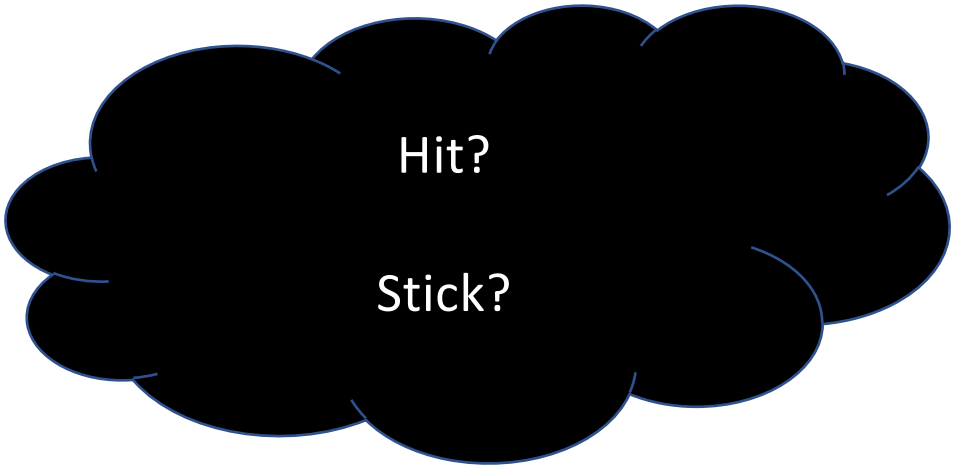
Monte Carlo

Value by Random Simulations

Monte carlo

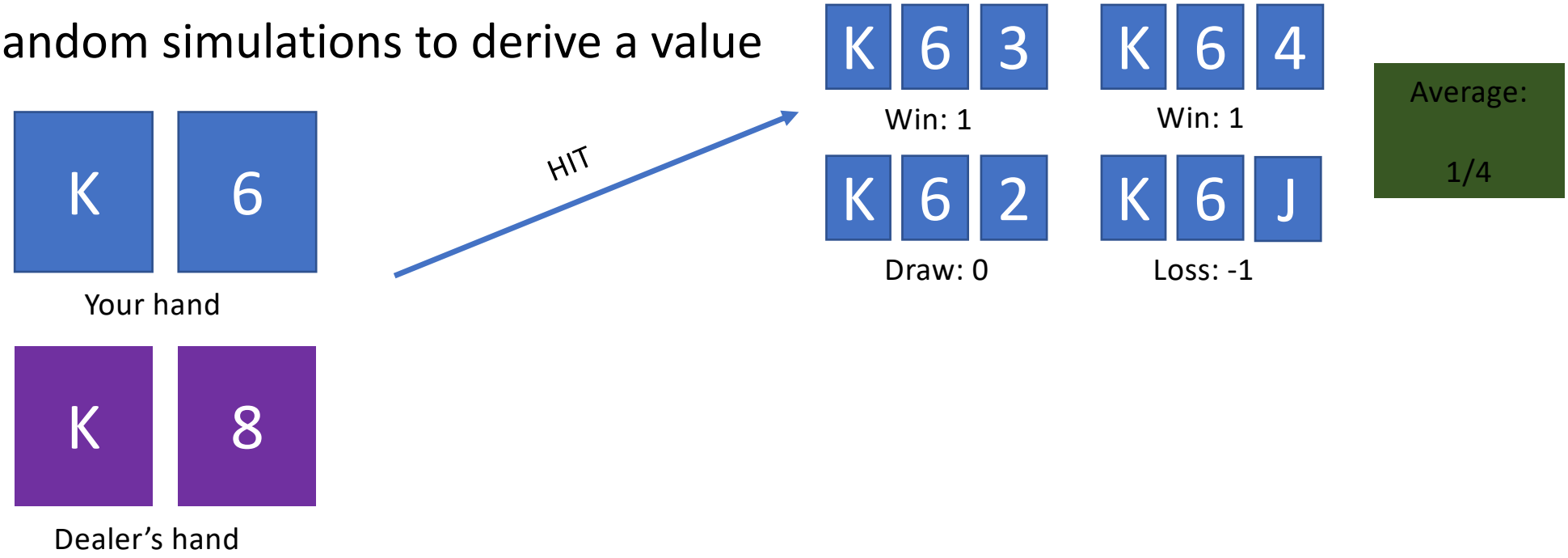
- Random simulations to derive a value





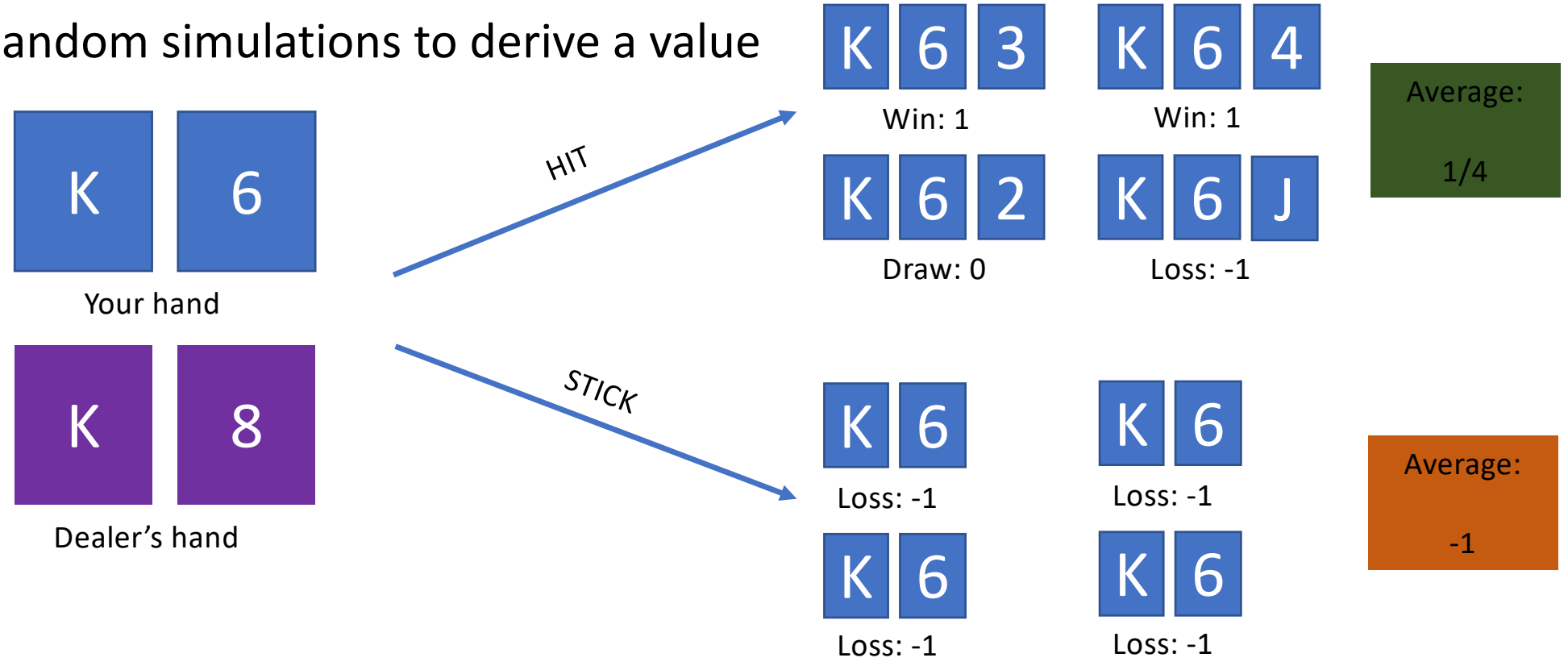
Monte carlo

- Random simulations to derive a value



Monte carlo

- Random simulations to derive a value



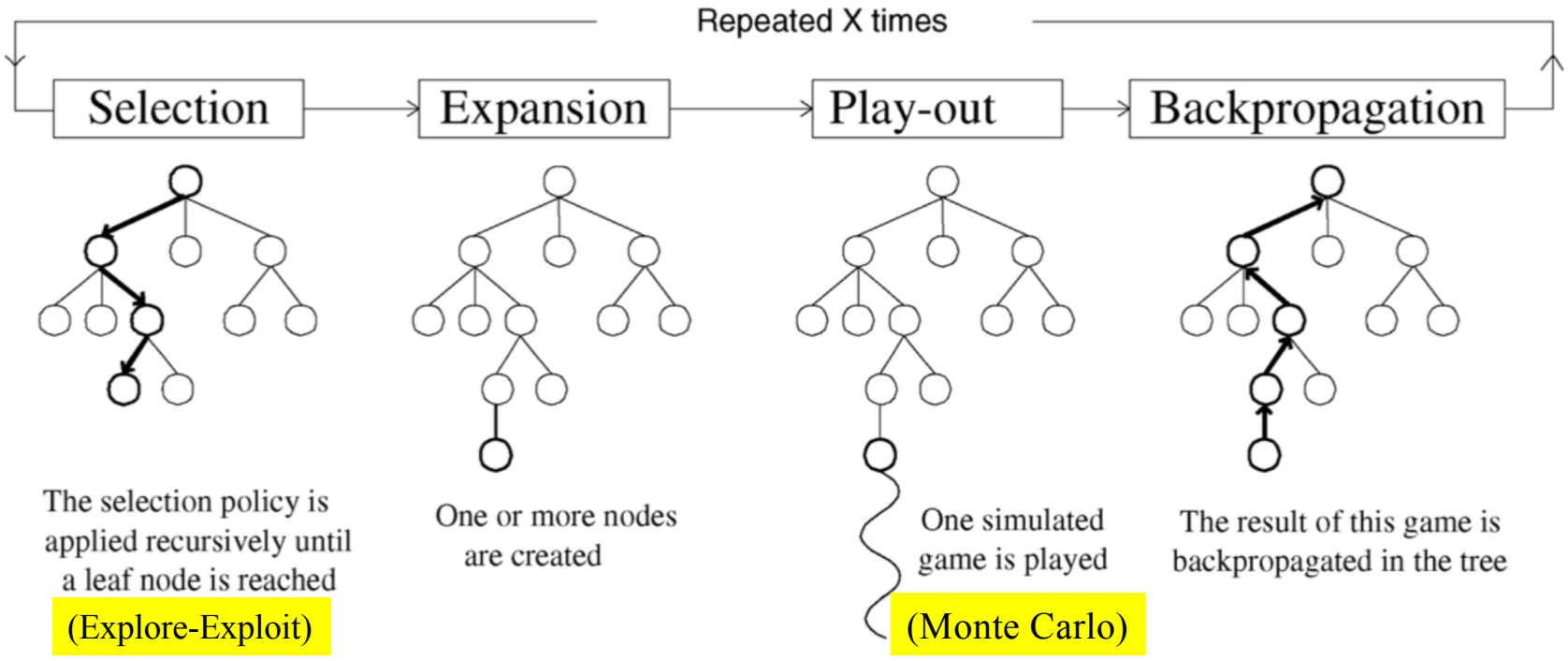
Monte Carlo Tree Search

Making sequential decisions

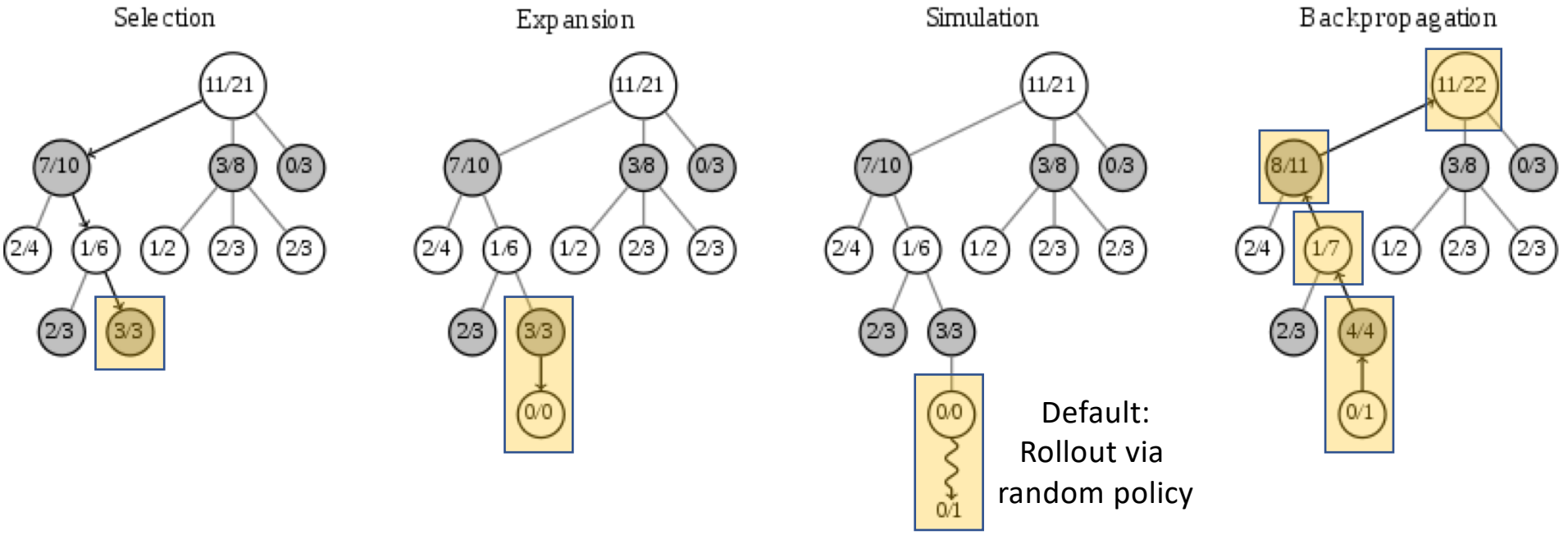
Select next state using Explore-Exploit

Approximate value of state using Monte Carlo

MCTS Procedure



MCTS Steps (Detailed)



Strengths

- Requires very little domain-specific knowledge: Only the rules
- Heuristic function to select nodes can incorporate domain knowledge
- Anytime algorithm

Limitations

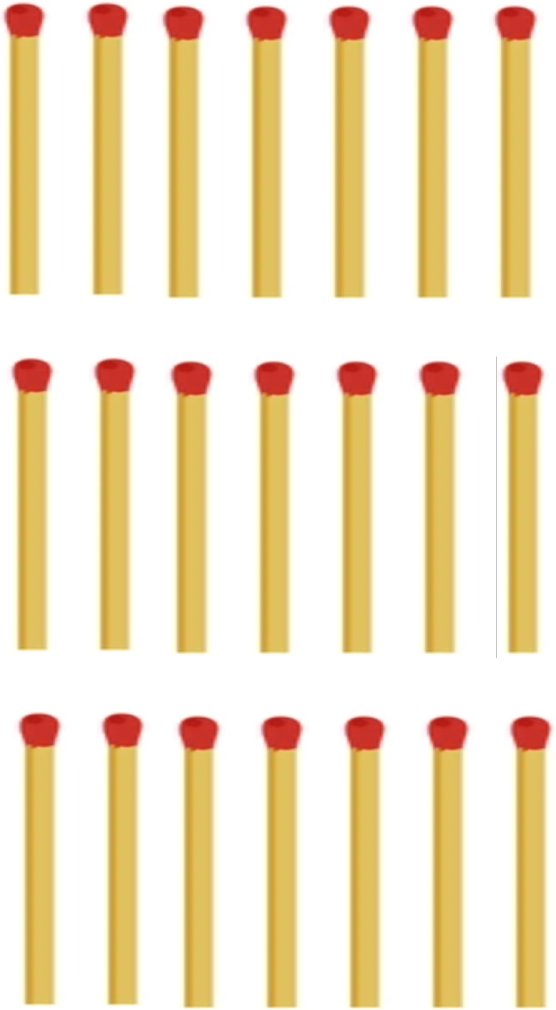
- Monte Carlo rollouts may not be an accurate gauge of state value
  - Long exact sequence to victory/loss may not be found
- Large amounts of simulations needed for larger state spaces
- How to solve?
  - Functional approximation via Neural Networks!
  - Heuristic encoding to select better moves
  - A form of TD-learning to improve value estimates

Practical Example

Solving the game of Nim with Monte Carlo / Monte Carlo Tree Search

Nim

- You have a pile of 21 matchsticks
- Each turn, you can pick 1 – 3 matchsticks (max is number of matchsticks remaining)
- Play rotates between players
- Player which picks the last matchstick loses



Jupyter Notebook

MC / MCTS with Nim