# Introduction to Machine Learning

## Reinforcement Learning

Varun Chandola

Computer Science & Engineering
State University of New York at Buffalo
Buffalo, NY, USA
chandola@buffalo.edu


University at Buffalo
Department of Computer Science and Engineering
School of Engeering and Applied Sciences

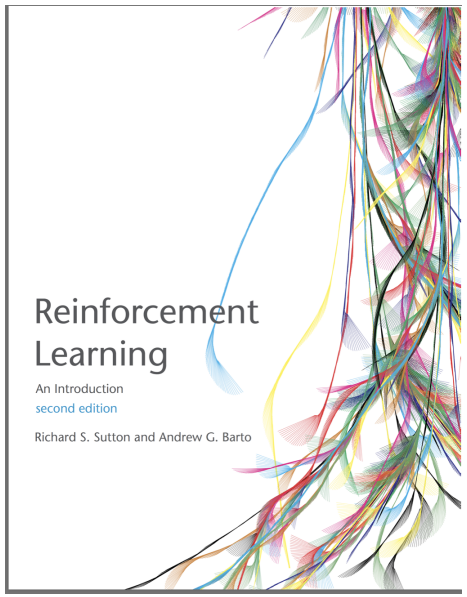
# Outline

# Introduction

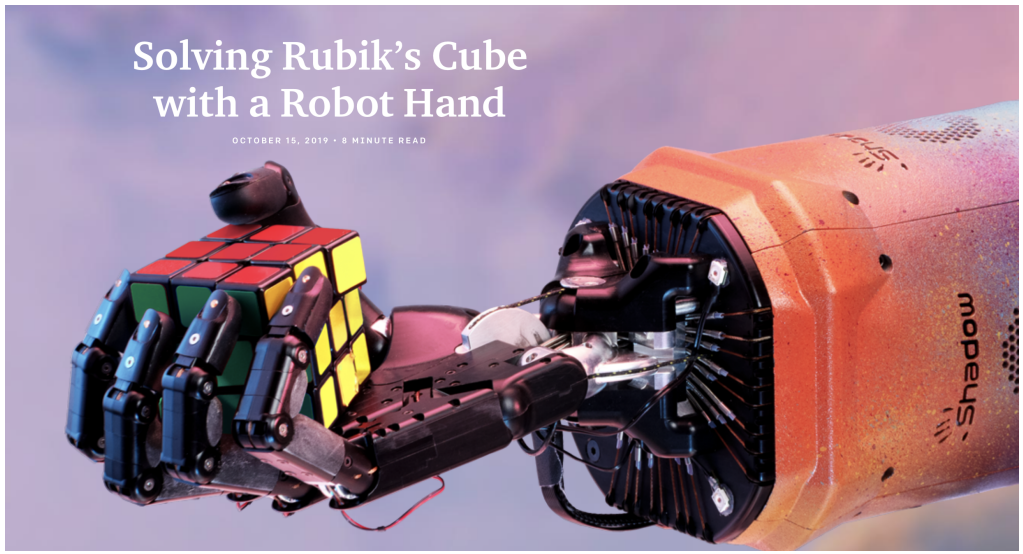
## Special Thanks to Alina Vereschaka

- CSE410/510 - Introduction to Reinforcement Learning

- Reinforcement Learning -Sutton and Barto



# What is Reinforcement Learning?




https://www.youtube.com/watch?v=x4O8pojMF0w

- Learn to take **actions** over a sequence of steps, to maximize **reward** over many time steps.

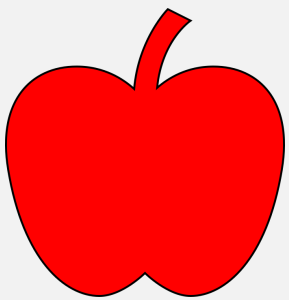
# Comparing all ML problems

## Supervised Learning

**Data**: $\langle \mathbf{x_i}, y_i \rangle$

**Task**: **Infer** $y^*$ for $\mathbf{x}^*$

**Example**:



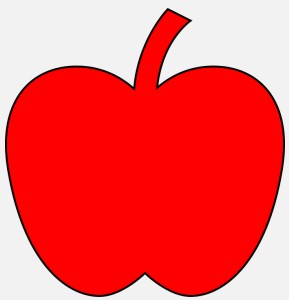
# Comparing all ML problems

## Supervised Learning

**Data**: $\langle \mathbf{x_i}, y_i \rangle$

**Task**: **Infer** $y^*$ for $\mathbf{x}^*$

**Example**:




## Unsupervised Learning

**Data**: $\langle \mathbf{x_i} \rangle$

**Task**: Learn **structure**

**Example**:

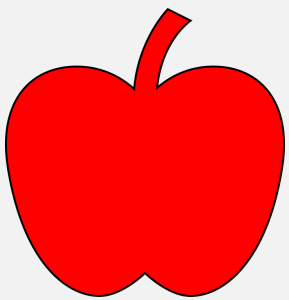
# Comparing all ML problems

## Supervised Learning

**Data**: $\langle \mathbf{x_i}, y_i \rangle$

**Task**: **Infer** $y^*$ for $\mathbf{x}^*$

**Example**:



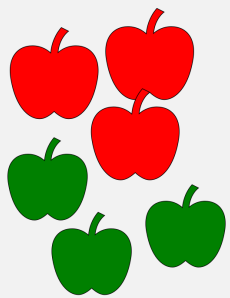
This is an apple

## Unsupervised Learning

**Data**: $\langle \mathbf{x_i} \rangle$

**Task**: Learn **structure**

**Example**:



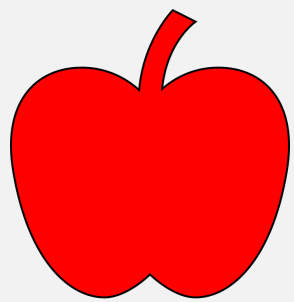
There are two types of apples

## Reinforcement Learning

**Data**: $\langle \text{state}, \text{action} \rangle$

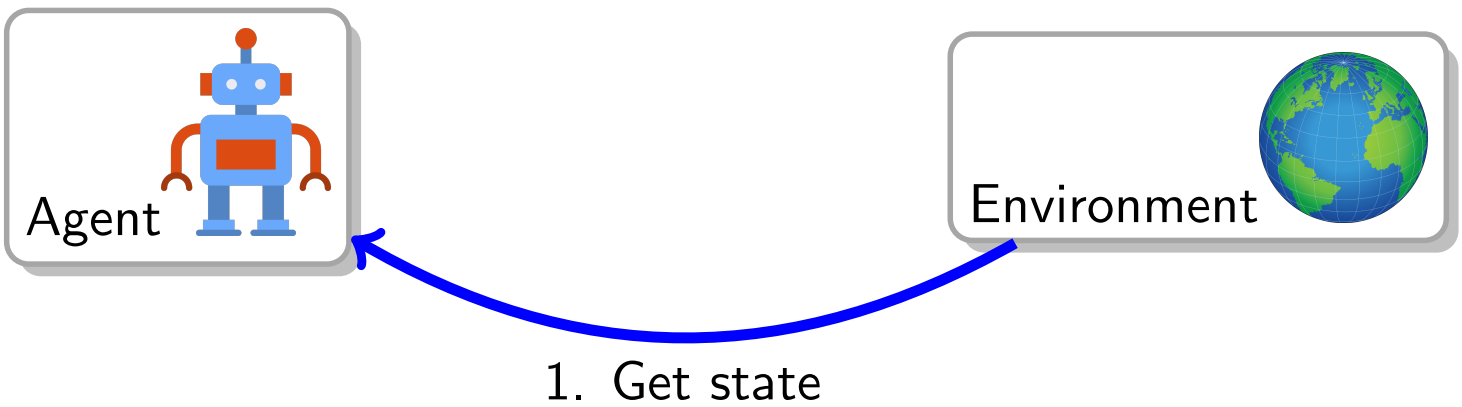
**Task**: Learn sequence of action to maximize reward

**Example**:

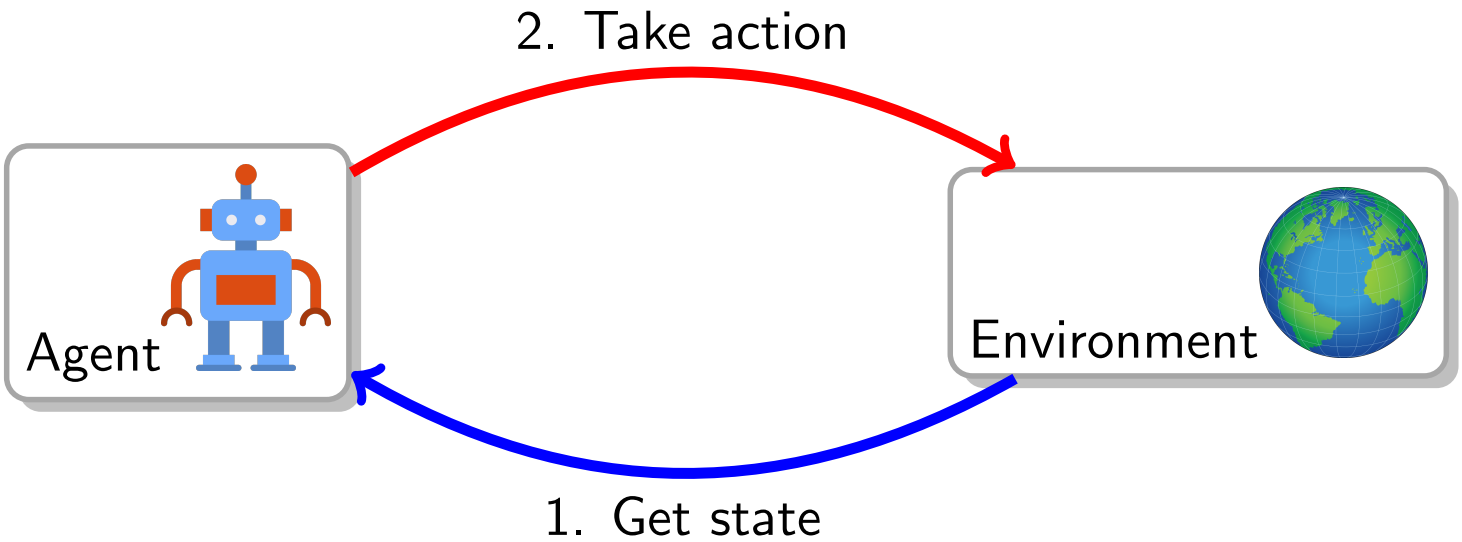


How to eat an apple

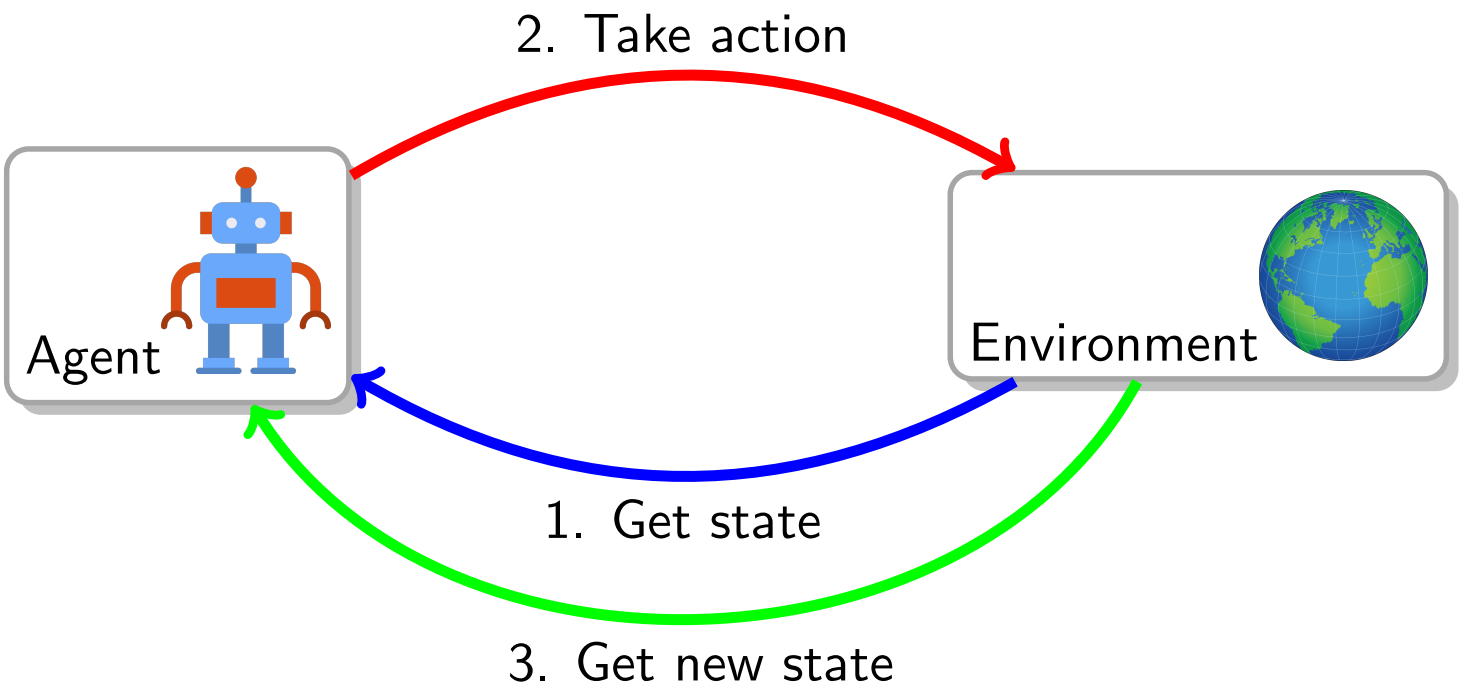
# Agent in an environment



# Agent in an environment



# Agent in an environment



# Examples

- Playing chess, Go, or many similar games
- Controller adjusting parameters of an engineered system (e.g., an oil refinery) in real time
- Robot learning to walk
- Everyday activities (e.g., making breakfast)

## What is common?

- Handling the *whole* problem of a goal-oriented agent in an uncertain environment
- Trade-off between *exploitation* and *exploration*

# Elements of a Reinforcement Learning Problem

- `Agent` operating in an `environment`
- `State` of the agent at time $t$ - $S_t \in \mathcal{S}$
- `Action` taken by agent at time $t$ - $A_t \in \mathcal{A}(S_t)$
- `Reward` at time $t$ - $R_t \in \mathcal{R}$
- `Policy` - $\pi$ (decision making rules)
  - $\pi(s) : \mathcal{S} \longrightarrow \mathcal{A}$
  - Action at a given state

## Goal of RL

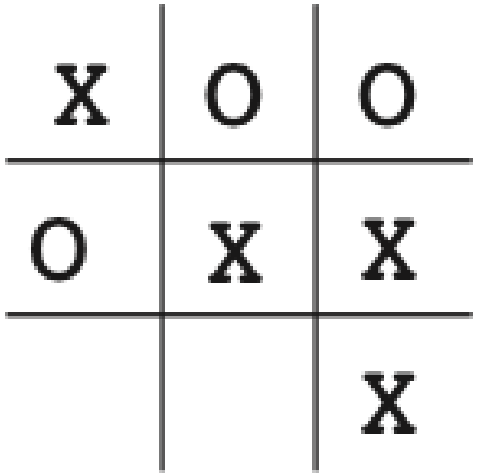
Learn optimal policy $\pi$ that maximizes the reward

# The Learning in Reinforcement Learning

1. `Policy`
2. `Reward signal` - used by the environment to inform the agent of the *reward* at a given time step
   - Primary basis for altering policy
   - Generally are functions of the current state and the actions
3. `Value function` - Expected total reward starting at a given state
   - Indicates the *long-term* desirability of a state
   - A state might have a small reward but a high value - might be followed by a sequence of states with high rewards
   - Harder to determine
4. `Model` - Allows us to model the environment (predict next states and next rewards)
   - Helps in planning

# Learning to play Tic-Tac-Toe

- Other player is the environment
- **Task**: Construct a player that maximizes the chance of winning

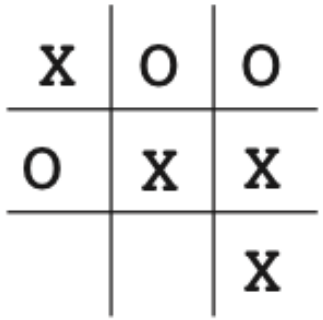



## Challenges

- Cannot assume a particular way of playing by the opponent
- Even then, you would need a completely specified model of the environment
- A possible approach – *learn* the model of the opponent's behavior by playing games against the opponent
- Or an exhaustive or evolutionary approach

# Learning Tic-Tac-Toe the RL way

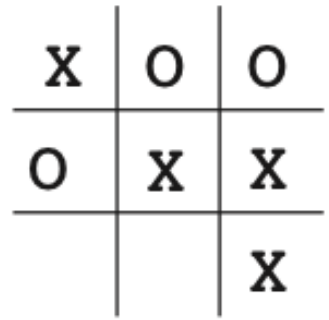
- State of the game is the configuration of $3 \times 3$ grid
- There can be $9^3$ possible states
  - Of course many are trivial




- Value of each state is the probability of winning from that state
- Set the value of the *obvious* states as 1, others 0.5
  - e.g., assuming we move $X$, the value of the shown state is 1
- Play many games with the opponent
  - With probability $(1 - \delta)$ choose a move that results in the highest value state (*exploitation*)
  - With probability $\delta$ choose a random move (*exploration*)

# Updating value

- After each greedy move, use the value of current state ($V(S_{t+1})$) to update the value of the previous state:




$$V(S_t) \leftarrow V(S_t) + \alpha[V(S_t + 1) - V(S_t)]$$

  - where $\alpha$ is a small positive fraction called the *step-size parameter*.
  - An example of a *temporal-difference* learning method

# Difference from evolutionary methods

## Evolutionary

- Operates at a policy level
- Evaluates a policy over many games and then chooses the next policy
- For each game, only the final outcome is considered

## Value function learning

- Evaluates inividual states during the course of play

# References