Introduction to Machine Learning

Reinforcement Learning

Varun Chandola

April 27, 2020

Outline

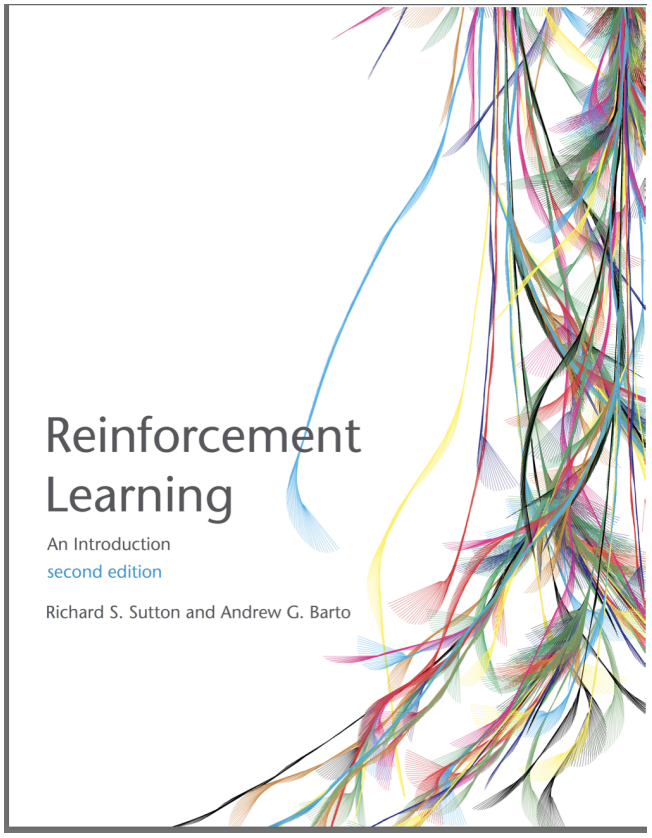
Contents

1 Introduction to Reinforcement Learning

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

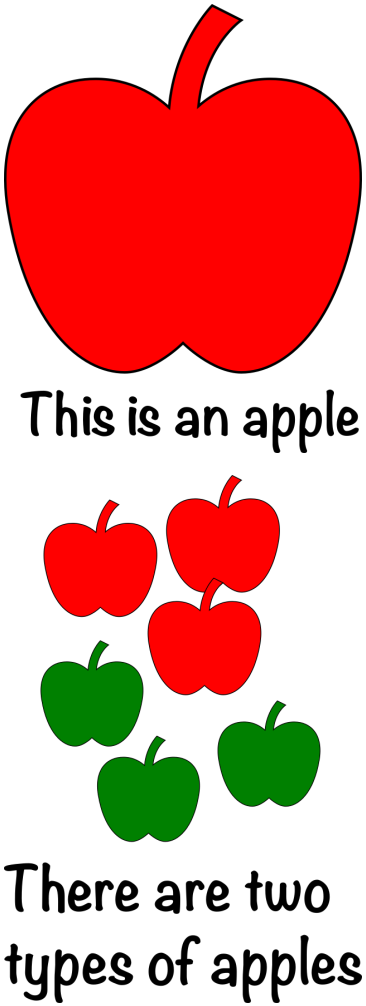
Supervised Learning
Data: $\langle \mathbf{x}_i, y_i \rangle$ **Task**: **Infer** y^* for $\mathbf{x}^*$ **Example**:

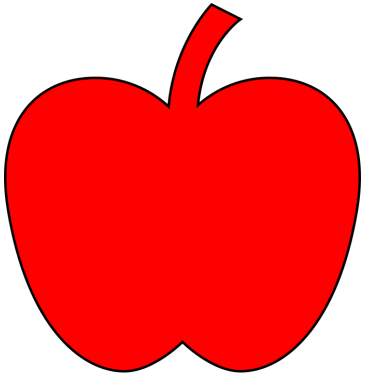
Unsupervised Learning
Data: $\langle \mathbf{x}_i \rangle$ **Task**: Learn **structure Example**:

Reinforcement Learning
Data: $\langle \text{state}, \text{action} \rangle$ **Task**: Learn sequence of action to maximize reward

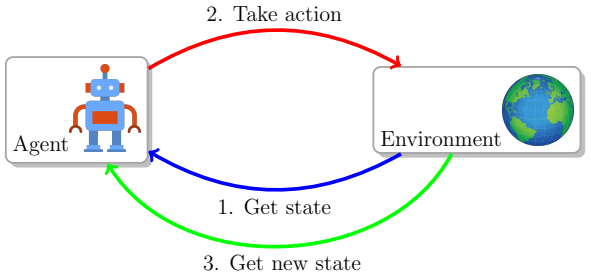
Example:

Agent in an environment





How to eat an apple



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* The agent can either exploit what it has already experienced or explore new actions that have not been considered before.

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` - π (decision making rules)
  - $\pi(s) : \mathcal{S} \longrightarrow \mathcal{A}$
  - Action at a given state

Goal of RL
Learn optimal policy π that maximizes the reward

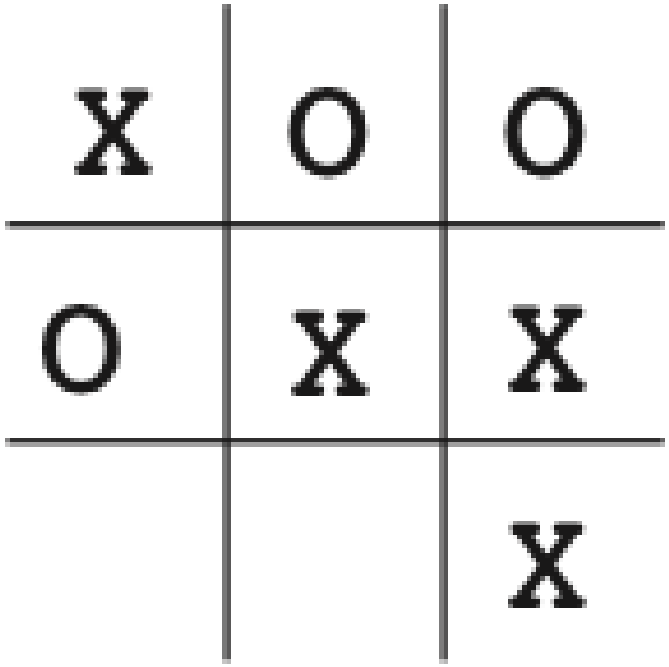
The Learning in Reinforcement Learning

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

1.1 Tic-Tac-Toe Example

Learning to play Tic-Tac-Toe

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



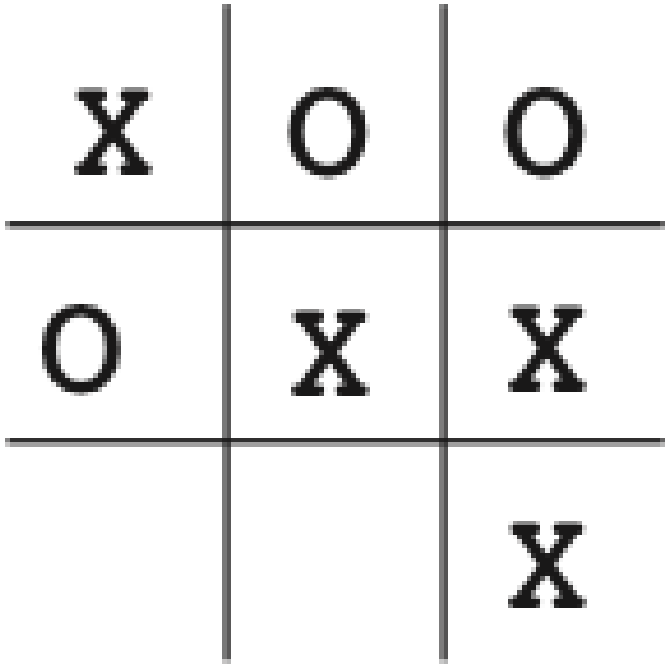
Challenges

- 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 Directly search the space of possible policies for one with a high probability of winning against the opponent. For each policy considered, an estimate of its winning probability would be obtained by playing some number of games against the opponent. Instead of exhaustively going through all possible policies, one could come up with an evolutionary strategy where the current policy would inform the choice of the next policy to evaluate.

Learning Tic-Tac-Toe the RL way

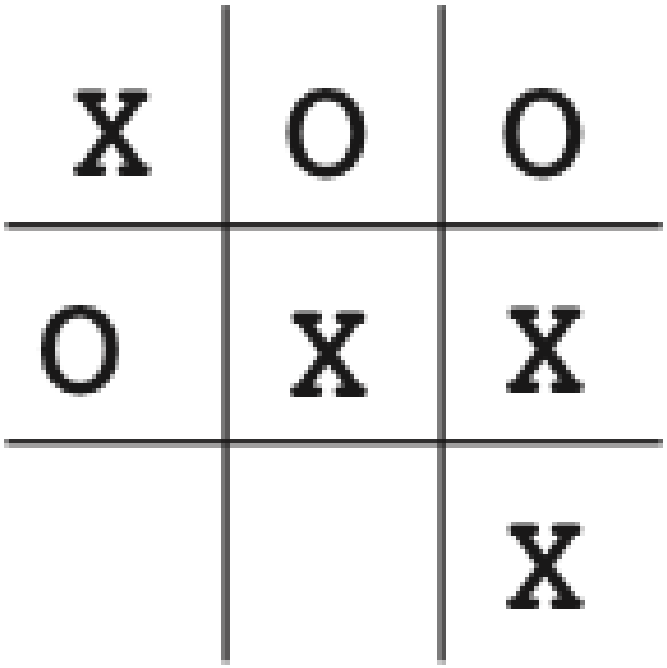
- State of the game is the configuration of 3×3 grid
- There can be 9^3 possible states
  - Of course many are trivial



- 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 δ 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 α 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

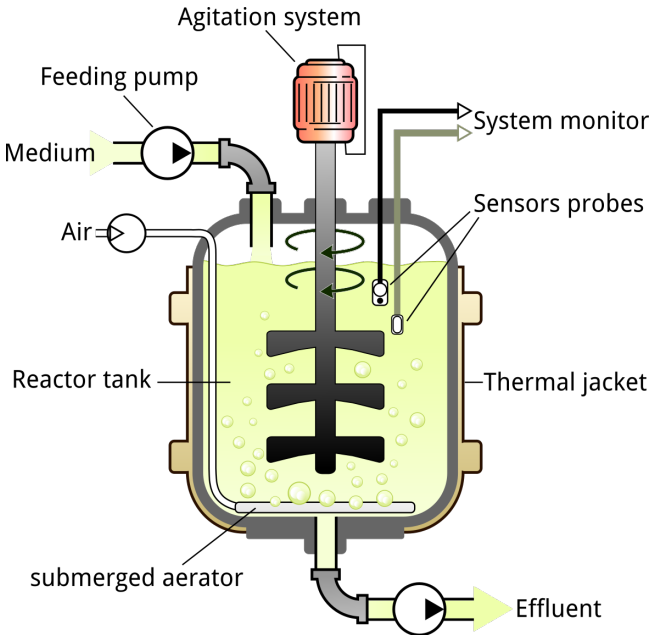
- Evaluates individual states during the course of play

Some Realistic Examples

- **Setting**: Bioreactor producing useful chemicals
- **Task**: Set optimal temperature and stirring rates during the operation

RL Setup

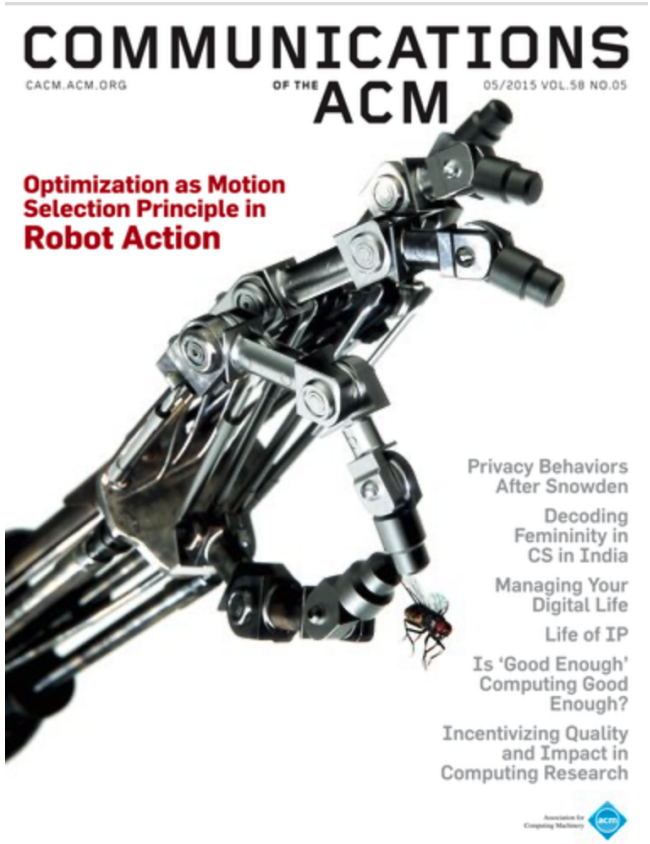
- **State**: Sensory readings (temperature, chemical composition)
- **Action**: Change temperature and/or stirring rate
- **Reward**: Measure of the useful chemical produced



- **Setting**: Robot picking up an object
- **Task**: Give optimal actuator inputs to the robot to enable smooth picking

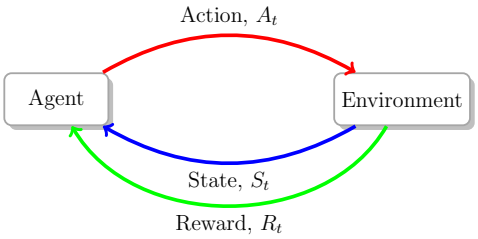
RL Setup

- **State**: Readings of joint angles and velocities
- **Action**: Change voltages to motors at each joint
- **Reward**: +1 if object picked up successfully
  - Additionally, one could give a small reward at each step if the motion is *not jerky*



2 Markov Decision Processes

Markov Decision Processes (MDP)



Environment state	$S_t \in \mathcal{S}$
Agent action	$A_t \in \mathcal{A}$
Reward	$R_t \in \mathcal{R}$

- A mathematically idealized form of RL
  - Can be analyzed theoretically
- Allows one to probabilistically reason about next state and reward, given the current state and action
- A wider range of RL applications can be formulated as finite MDPs
- But there are other ways beyond MDP

Markov Decision Processes (MDP)

- The agent-environment interaction gives rise to a sequence or *trajectory*:

$$S_0, A_0, R_1, S_1, A_1, R_2, S_2, A_2, R_3, \ldots$$

- Finite MDP assumes that $\mathcal{S}, \mathcal{A}, \mathcal{R}$ are finite
- A joint probability distribution can be defined for (s', r), where $s' \in \mathcal{S}$ and $r \in \mathcal{R}$:

$$p(s', r|s, a) \doteq P(S_t = s', R_t = r|S_{t-1} = s, A_{t-1} = a)$$

  for all $s', s \in \mathcal{S}, r \in \mathcal{R}, a \in \mathcal{A}(s)$
- p defines the *dynamics* of the MDP
  - A four argument function: $\mathcal{S} \times \mathcal{R} \times \mathcal{S} \times \mathcal{A} \to [0, 1]$

Why *Markovian?*

- The probabilities given by p completely characterizes the environment's dynamics
  - Probability of each possible value for S_t and R_t depends on the S_{t-1} and A_{t-1}, and nothing before

Versatility of dynamics function p

- Reveals "everything" about the environment:
  1. State-transition probabilities:
  $$p(s'|s,a) \doteq \sum_{r\in\mathcal{R}} p(s',r|s,a)$$
  2. Expected reward for a given state-action pair:
  $$r(s,a) \doteq \mathbb{E}[R_t|S_{t-1}=s, A_{t-1}=a] = \sum_{r\in\mathcal{R}} r \sum_{s'\in\mathcal{S}} p(s',r|s,a)$$
  3. Expected rewards for state−action−next-state triplet:
  $$r(s,a,s') \doteq \mathbb{E}[R_t|S_{t-1}=s, A_{t-1}=a, S_t=s'] = \sum_{r\in\mathcal{R}} r\frac{p(s',r|s,a)}{p(s'|s,a)}$$

Two types of tasks

Episodic Tasks

- Agent-environment interaction can be broken into subsequences or *episodes*.
  - Plays of a game, trips through a maze, or any other repeated interaction
- Each episode ends in a terminal states
- Next episode begins independently of how the previous one ended
- $\mathcal{S}$ is usually used to denote set of all *non-terminal* states

- $\mathcal{S}^+$ denotes the set of all states (including non-terminal)

Continuing Tasks

- Agent-environment interaction cannot be naturally broken into identifiable episodes
  - Any life-long learning task

Formal Definition of Learning Goal

- Maximize the *expected return*

Expected Return

$$G_t \doteq R_{t+1} + R_{t+2} + \ldots + R_T$$

where T is the final time step.

- The above formulation does not work for *continuing tasks* where $T = \infty$

Expected Return with Discounting

$$G_t \doteq R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+3} \ldots + R_T$$

where γ is the *discount rate parameter*, $0 \leq \gamma \leq 1$.

Significance of the discount rate

- If $\gamma = 0$, the agent is only maximizing the immediate reward (short-sighted)
- A γ tends to 1, the agent gives more weight to future rewards
- Connection between successive expected returns:

$$G_t = R_{t+1} + \gamma G_{t+1}$$

References