Introduction to Reinforcement Learning

Weizi Li

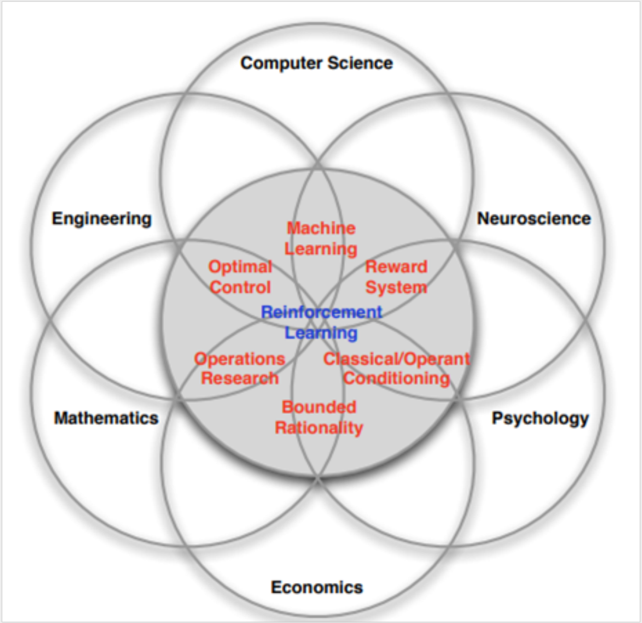
Department of Computer Science
University of Memphis

Lecture Outline

- What is RL
- Reward
- RL Agent
- Brief History of RL

What is RL

Sequential Decision Making

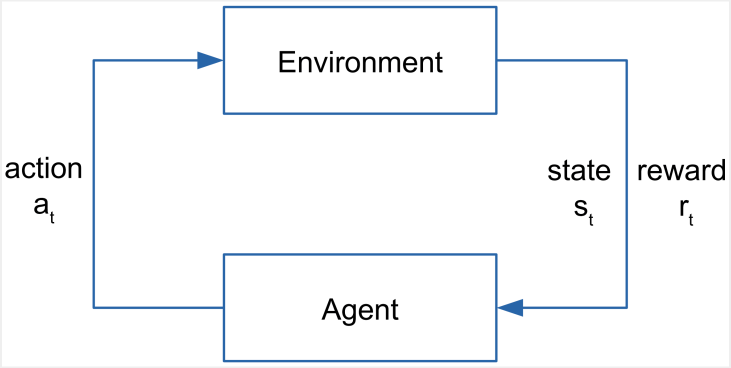


Example Applications

- Computer science: games, robotics...
- Operations: inventories, production, revenue management...
  - Manufacturing: choose a sequence of actions to maximize profits.
  - Taxi scheduling: pick up passengers in a city to minimize waiting time.
- Finance: portfolio management, options...
- Control: process control, path planning, robotics...

Reinforcement Learning Problem

- How an autonomous agent that senses and acts in its environment can learn to choose optimal actions sequentially to maximize its reward



Recent Advancements

- 2015: Atari games superhuman on some games
- 2016: Game of Go (AlphaGo) superhuman
- 2017: Game of Go and Chess (AlphaZero) superhuman
- 2018: Robot manipulation (real robots)
- 2019: Dota 2 (OpenAI Five) superhuman
- 2019: StarCraft II (AlphaStar) grandmaster
- 2019: Solving Rubik's cube with a robot hand (real robots)
- 2021: Atari games (Go-Explore) superhuman on all games
- 2021: Chip floor planning superhuman
- 2022: Gran Turismo 7 superhuman

Key Concept: Agent

- Agent can sense its environment and choose actions to execute in the environment

Key Concept: Environment

- The world that the agent lives in
  - fully observable vs. partially observable
  - single-agent vs. multiagent
  - deterministic vs. nondeterministic
  - known vs. unknown

Agent-Environment Boundary

- Agent: the decision-making process
- Environment: the "physical" components
- The agent–environment boundary represents the limit of the agent's absolute control
- For convenience, people use the term agent loosely, e.g., we call robot agent even its physical components belongs to the environment

Key Concept: Action

- The movement or other types of decisions available to the agent
- An executed action will affect the environment
- An action may have a long-term consequence
- Choosing an optimal action is difficult: make decisions beyond the immediate payoff

Key Concepts: State

- A snapshot of the environment at a particular point in time
- Fully observed: agent observes environment fully
- Partially observed: agent observes environment partially (e.g., robot with a front-facing camera)
- May contain irrelevant information for a specific task
- Information in state may be perturbed due to flaws of sensor/processing algorithms

Key Concept: Reward

- A numerical value (scalar)
- An immediate payoff to each distinct action that agent chooses to take in each distinct state
- Can be 0 (no reward) or even -1 (penalty) at intermediate steps of many tasks
- The actual reward can be <u>sparse</u> and <u>delayed</u>

Key Concept: Reward Function

- A function executed by the environment to determine the immediate payoff after agent executes an action in a state

Key Concept: Transition Function

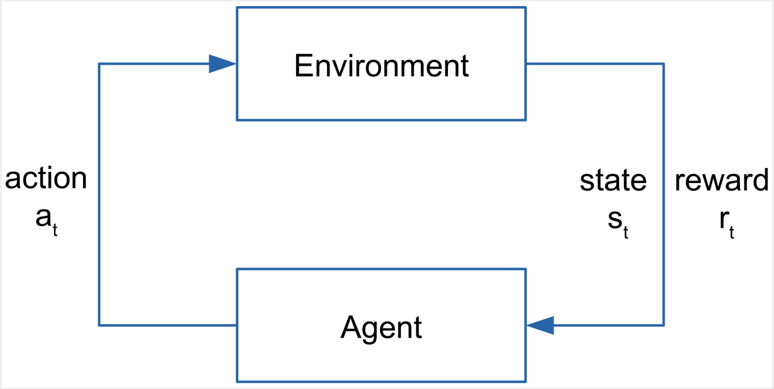
- A function executed by the environment to determine the next state after agent executes an action in the current state

Key Concept: Policy

- Defines the behavior of an agent by mapping states to actions
- The optimal policy is one that, from any initial state, chooses actions that maximize the reward accumulated over time

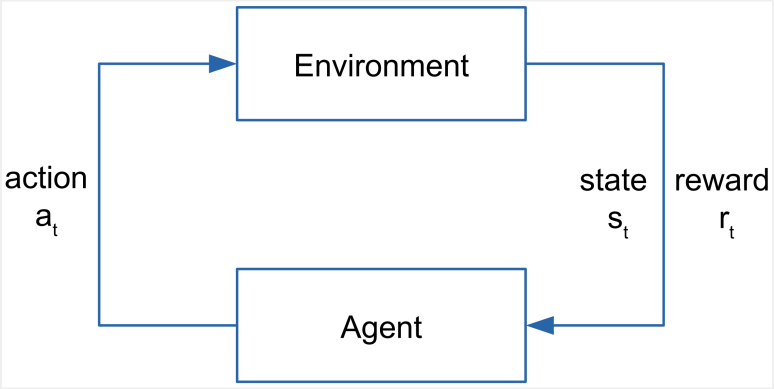
RL Pipeline

- Agent (via a policy) chooses an action to execute in the current state



RL Pipeline

- Environment launches its reward function and transition function to produce immediate payoff to agent and transit agent to the next state



Goal of An RL Agent

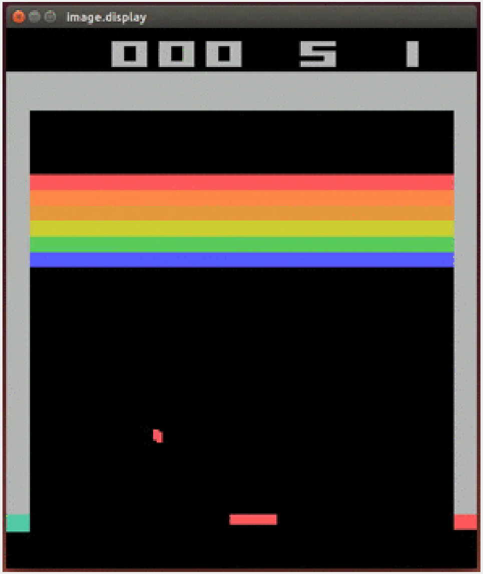
- Perform sequences of actions, observe their consequences (i.e., rewards), and learn an optimal policy
- During learning, agent does not need any prior knowledge of the effects of its actions on the environment

Compare to Supervised Learning

- Supervised learning: recognition problem
- RL: decision problem
- Learn via trial and error, no supervisor only indirect, delayed feedback
- Action order (time) matters, non-i.i.d. data
- More difficult learning task: usually learn fewer parameters (dozes and thousands) than supervised learning (millions) due to limited feedback

Example: Atari Game

- Agent
- Action
- Environment
- State
- Reward

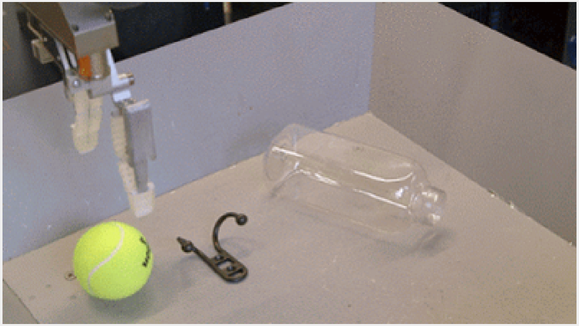


Example: Go

- Agent
- Action
- Environment
- State
- Reward

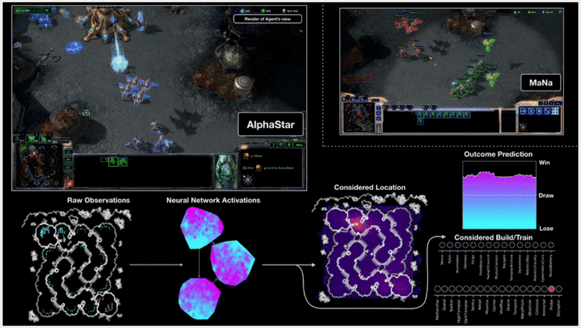
Example: Robot Arm

- Agent
- Action
- Environment
- State
- Reward



Example: StarCraft

- Agent
- Action
- Environment
- State
- Reward



Reward

Terminology

- "reward / immediate reward / indirect reward": immediate payoff at each timestep, could be "no reward" or "penality"

Terminology

- “cumulative reward”: sum of reward at each timestep

Terminology

- “delayed reward / sparse reward”: reward you actually want

Reward Hypothesis of RL

- The goal of RL agent is maximizing expected cumulative reward

Reward Design

- Manually designed reward can be brittle (link)

Reward Design

- Reward can be difficult to define, e.g., how to define “good driving” using a number?

Reward Design

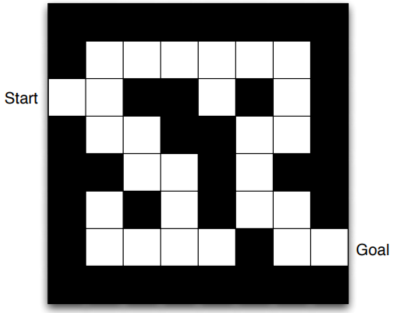
- Where does reward even come from?
- Human preferences? “Deep reinforcement learning from human preferences” by Christiano et al, 2017
- As a general rule, it’s better to design performance measures according to what one actually wants to be achieved in the environment, rather than according to how one thinks the agent should behave. Safety -> car in parked in garage

RL Agent

Components of an RL Agent

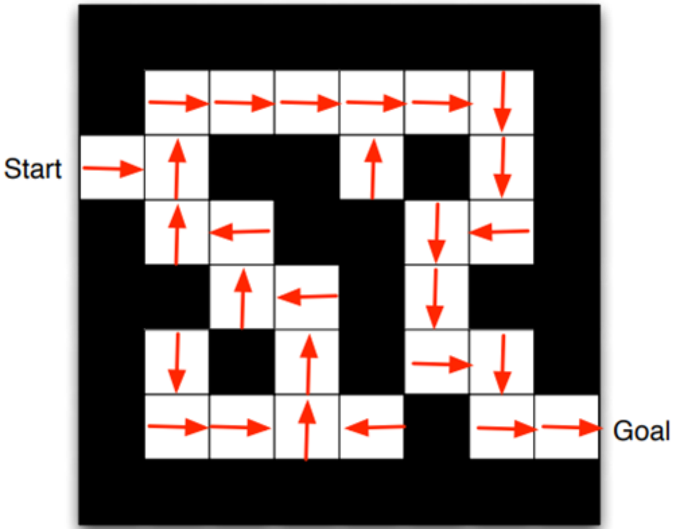
- Policy
  - deterministic: $\pi(S) = A$
  - stochastic: $\pi(A|S) = P[A_t = A|S_t = S]$
- Value function
  - estimated total future reward starting from a state or a state-action pair
  - help choose action (implicit policy)
- Model (optional)
  - agent's understanding of the environment dynamics (i.e., reward function and transition function)

Maze

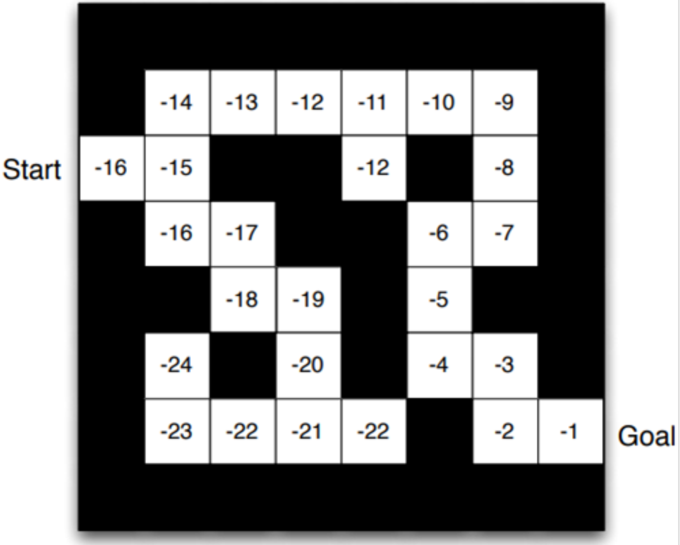


- Rewards: -1 per time-step
- Actions: N, E, S, W
- States: Agent's location

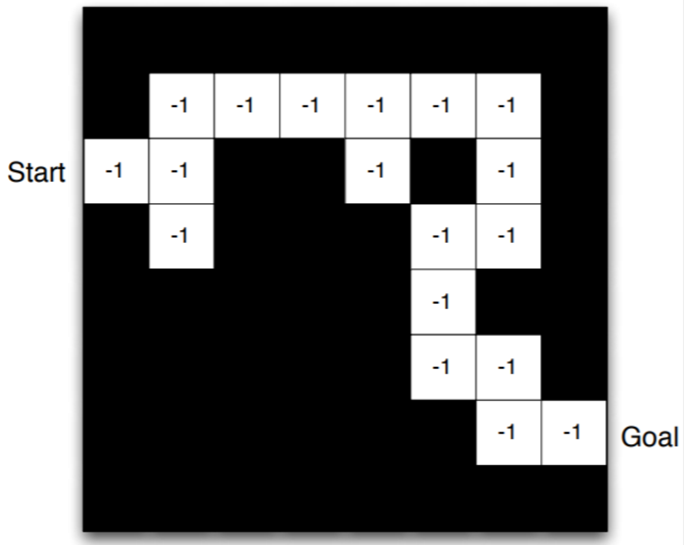
Policy



Value Function



Model



Model-based vs. Model-free

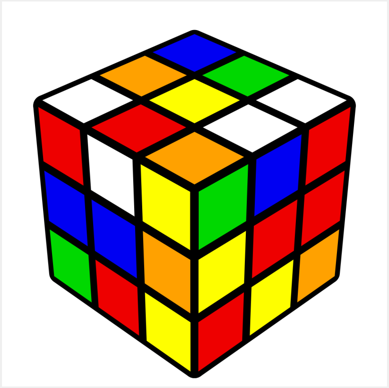
- Model-based learning: building models of the environment, e.g., compute expectation by first estimate event probabilities and then multiply them with the corresponding function values.

$$E[f(x)] \approx \sum \hat{p}(x) f(x).$$

- Model-free learning: do not build models of the environment, e.g., compute expectation as the average of the sampled function values:

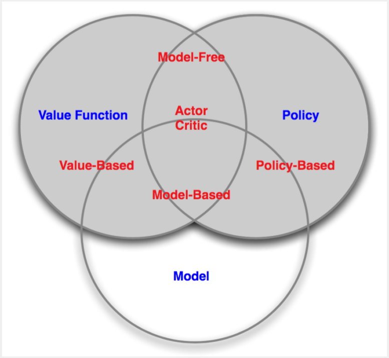
$$E[f(x)] \approx \frac{1}{n} \sum_{i=1}^{n} f(x_i).$$

Knowing Models $\neq$ Easy Problem



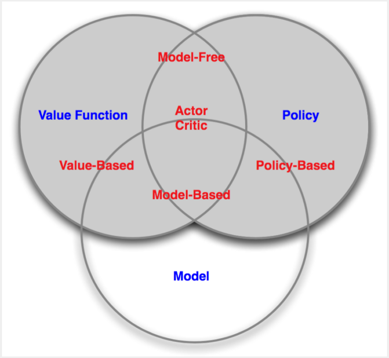
Types of RL Agent

- Model-based
    - Value function and/or policy
- Model-free
    - Value function and/or policy



Types of RL Agent

- Value-based
  - Value function (no implicit policy)
- Policy-based
  - Policy (no value function)
- Actor-critic
  - Value function and Policy



RL in One Sentence

- Learn optimal behavior via trial and error, based on environment's feedback

Brief History of RL

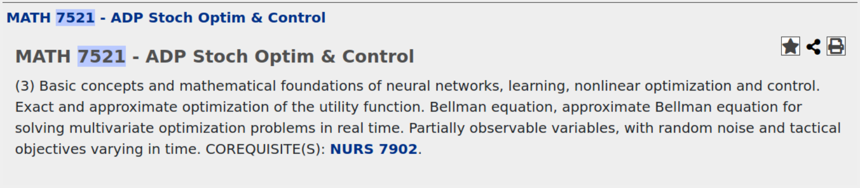
RL Threads

- Three threads joined in 80's to form the modern RL
    - Thread 1: animal learning (psychology)
    - Thread 2: optimal control (engineering)
    - Thread 3: artificial intelligence (computer science)

Control Thread

- “Dynamic Programming” by Bellman, 1957
- “Dynamic Programming and Stochastic Control” by Bertsekas, 1976
- “Neuro-Dynamic Programming” by Bertsekas and Tsitsiklis, 1996
  - Then “approximate dynamic programming”
  - Now “deep reinforcement learning”

Control Thread


MATH 7521 - ADP Stoch Optim & Control

MATH 7521 - ADP Stoch Optim & Control

(3) Basic concepts and mathematical foundations of neural networks, learning, nonlinear optimization and control. Exact and approximate optimization of the utility function. Bellman equation, approximate Bellman equation for solving multivariate optimization problems in real time. Partially observable variables, with random noise and tactical objectives varying in time. COREQUISITE(S): **NURS 7902**.

AI Thread

- World's first successful checkers-playing program by Arthur Samuel, 1959
- TD-GAMMON by Tesauro (among the world's best backgammon players), 1992
- Machine intelligence based on animal learning psychology by Harry Klopf, 1970's
- Temporal-difference learning, 1980's
- "Reinforcement Learning: An Introduction" by Sutton and Barto, 1998, (2nd ed. in 2020)

New Wave

- Breakthroughs from DeepMind, OpenAI, etc.: Atari games (2015), AlphaGo (2016)...
- Mostly due to increased computing power + engineering practices, not much on the theory side
- Richard Bellman: “Once working in the area, it is very quickly realized that far more ability and sophistication is required to obtain a numerical solution than to establish the usual existence and uniqueness theorems”