Foundations of Reinforcement Learning

Alessio Rimoldi

March 14, 2025

Chapter 1

Introduction

1.1 What is Reinforcement Learning?

Reinforcement Learning (RL) is an area of machine learning concerned with how intelligent agents ought to take actions in an environment in order to maximize some notion of cumulative reward. The agent learns to achieve a goal in an uncertain, potentially complex environment. In RL, an agent interacts with its environment, observes the state of the environment, and takes actions that affect the state. he agent also receives rewards from the environment. The agent's goal is to learn to act in a way that will maximize its expected cumulative reward over time.

1.1.1 People and Connected Fields

- Computer Science: Minsky, Barto, Sutton
- Neuroscience:
- Psychology: Holland, Klopf
- Operations Research: Bertsekas, Puterman
- Economics:
- Engineering: Bellman, Tsitsiklis

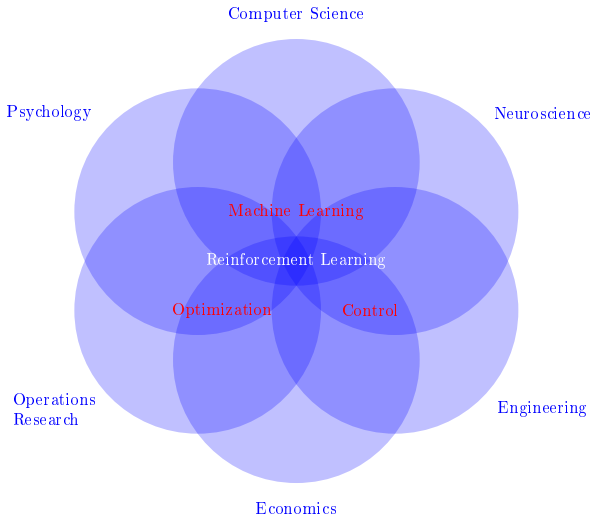

Figure 1.1: Reinforcement Learning

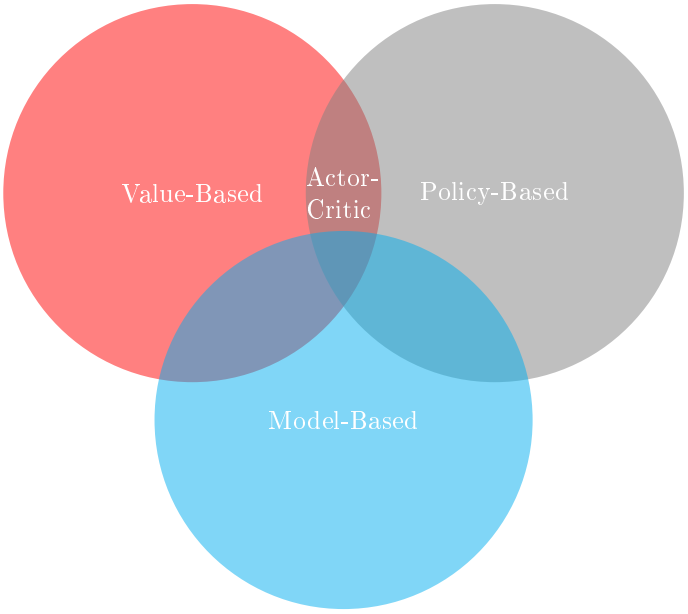

Figure 1.2: Reinforcement Learning Approaches

1.1.2 Overview of Reinforcement Learning Approaches

- **Value-based**
  - Learns the optimal value function
  - Example: Monte Carlo, SARSA, Q-learning, DQN
  - Low variance, not scalable to large action spaces
- **Policy-based**
  - Learns directly the optimal policy
  - Example: Policy Gradient, NPG, TRPO, People
  - Scales to lathe and continuous action spaces, high variance
- **Model-based**
  - Learns both the model state transition probability , reward and the optimal Policy
  - Example: Dyna, UCRL2, UCB-variance
  - Computationally expensive, but better sample efficiency

Chapter 2

Preliminaries

Chapter 3

Value-Based Methods

Fundamental Challenge 1

The dynamic programming approaches, Value Iteration and Policy Iteration, and the Linear Programming approach all require full knowledge of the model transition probabilities P and the reward function r.

Solution: Learning

Fundamental Challenge 2

The computation and memory cost can be very expensive for large scale Markov Decision Process (MDP) problems.
Solution: Representation = Function Approximation

3.0.1 Function Approximation Methods

- **Linear Function Approximation**
  - Linear combination of basis functions
  - reproducing kernel Hilbert space
  - Neural tangent kernel
- **Non-Linear Function Approximation**
  - Fully connected Neural Networks
  - Convolutional Neural Networks
  - Recurrent Neural Networks
  - Self attention
  - Generative Adversarial Networks

3.0.2 Model-free Prediction

Given a policy $\pi : \mathcal{S} \to \Delta(\mathcal{A})$, estimate $V^\pi(s)$ pr $Q^\pi(s,a)$ from episodes of experience unde π.

$$V^\pi(s) := \mathbb{E}\left[\sum_{t=0}^{\infty} \gamma^t r(s,a) | s_0 = s\right]$$

We also know from the Bellman consistency equation that

$$V^\pi(s) = \mathbb{E}_{a\sim\pi(\cdot|s)}\Big[\sum_{t=0}^{\infty} \gamma^t r(s_t,a_t) | s_0 = s, \pi\Big]$$

3.1 Monte Carlo Method

Idea: Estimate $V^\pi(s)$ by the average of returns following all the visits to s.

Monte Carlo Method

for each episode **do**
    Generate an episode $\{s_0, a_0, r_0, s_1, \dots\}$ following π
    for each state s_t **do**
        Compute return $G_t = r_{t+1} + \gamma r_{t+2} + \dots$
        Update counter $n_t(s_t) \leftarrow n_t(s_t) + 1$
        Update $V(s_t) \leftarrow V(s_t) + \frac{1}{n_t}(G_t - V(s_t))$
    end for
end for

- Value estimates are **independent** and do not build on the ones of other states. (**no bootstrap**)
- Learning can be slow when the episodes are long.
- **Convergence:** Monte Carlo (MC) converges to $V^\pi(s)^p i$ as $n_t(s_t) \to \infty$.

3.2 Temporal Difference Learning

Idea: Incrementally estimate $V^\pi(s)$ by the intermediate returns plus estimated return at the next state

$$V(s_t) \leftarrow V(s_t) + \alpha_t(r_{t+1} + \gamma V(s_{t+1}) - V(s_t)) r_{t+1} + \gamma V(s_{t+1}) - V(s_t) = \delta_t \quad \text{Is the TD error}$$

TD Learning / TD(0)

for each step **do**
    Observe (s_t, a_t, r_t, s_{t+1})
    Update $V(s_t) \leftarrow V(s_t) + \alpha_t(r_{t+1} + \gamma V(s_{t+1}) - V(s_t))$
end for

- Similar to Dynamic Programming (DP): the estimates build on estimates of the other state (**bootstrap**)
- Similar to MC: learn directly from episodes of the experience without knowledge of MDP.
- Unlike MC, can learn from incomplete episodes, applicable to non-terminating environment.
- **Convergence:** $V \to V^\pi$ if each state is visited infinitely often and $\alpha_t \to 0$ at a suitable rate.

Chapter 4

Policy Gradient Methods