RL Introduction

Ruichen Wang

July 22, 2020

Abstract

AI=RL+DL

Contents

1 Basis

1.1 Introduction

Reinforcement learning is learning what to do, how to map situations to actions, so as to maximize a numerical reward signal. The learner must discover which actions yield the most reward

by trying them.

Of all the forms of machine learning, reinforcement learning is the closest to the kind of learning that humans and other animals do, and many of the core algorithms of reinforcement learning were originally inspired by biological learning systems.

In the most interesting and challenging cases, actions may affect not only the immediate reward but also the next situation and, through that, all subsequent rewards.

Two most important distinguishing features:

- Trial and error search
- Delayed reward

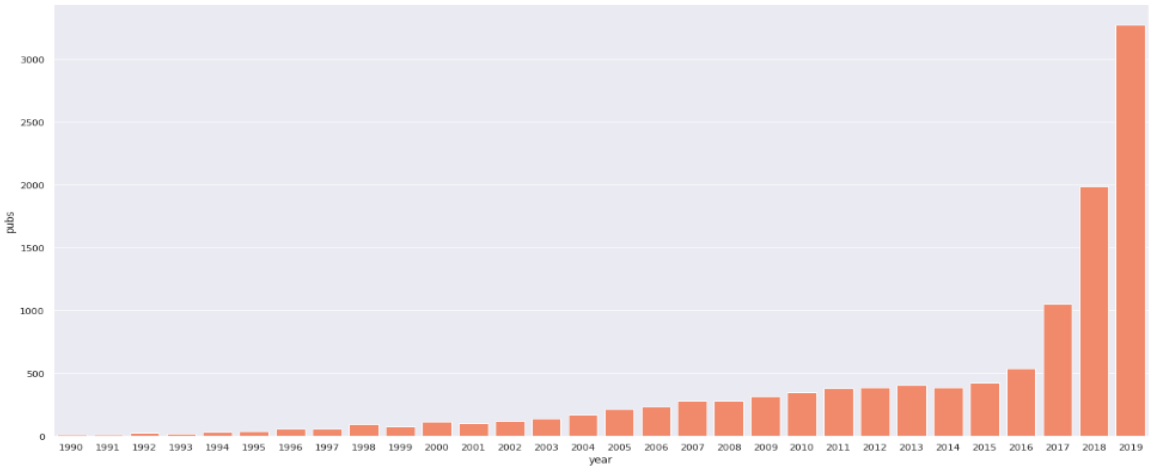

Figure 1: Publications by year

1.2 Formalize Definition

We formalize the problem of reinforcement learning using ideas from **dynamical systems theory**, specifically, as the optimal control of **incompletely-known Markov decision processes**.

Three key features:

- Sensation : sense the state of its environment to some extent
- Action : take actions that affect the state
- Goal : a goal or goals relating to the state of the environment

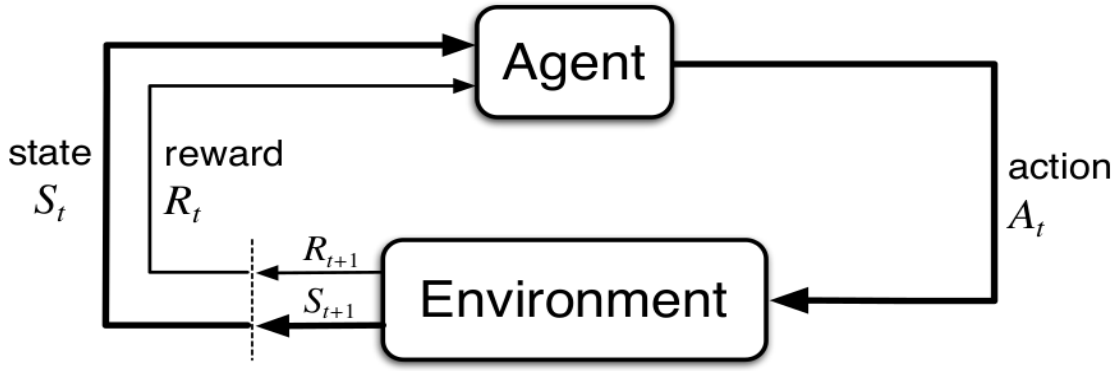

Figure 2: MDP

1.3 Difference Between Supervised or Unsupervised Learning

In supervised learning, we have training set and labeled examples. In interactive problems it is often impractical to obtain examples of correct and representative behavior. An agent must be able to learn from its own experience.

In unsupervised learning, we aim to find structure hidden in collections of unlabeled data. Reinforcement learning is aiming to maximize a reward signal.

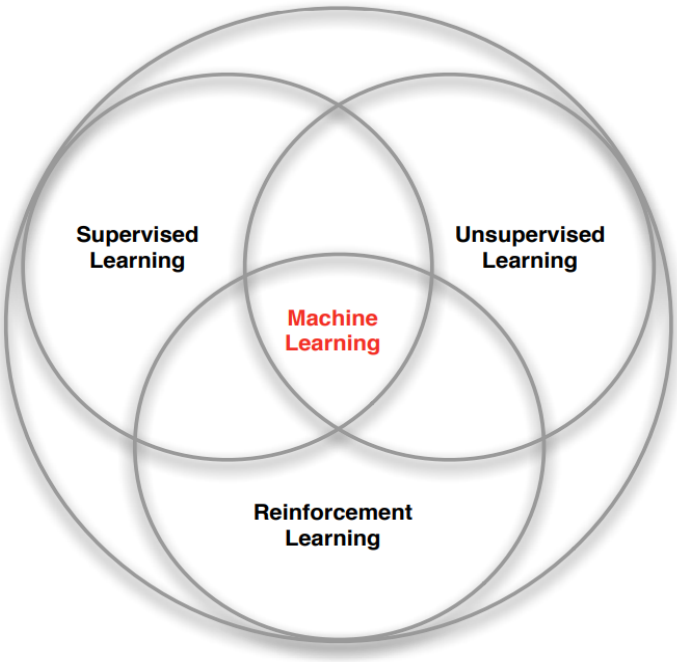

Figure 3: Relationship

1.4 Trade-off between Exploration and Exploitation

The dilemma is that neither exploration nor exploitation can be pursued exclusively. We want the agent should be able to discover good policy without losing too much reward. Same important.

2 Elements of Reinforcement Learning

Beyond the **agent** and the **environment**, one can identify four main subelements of a reinforcement learning system: a **policy**, a **reward signal** , a **value function**, and, optionally, a **model of the environment**.

2.1 Environment

Environment can be classfied by whether its fully observable(MDP) or partially observable(POMDP).

2.1.1 State, Observation and History

A state S is a complete description of the state of the world. There is no information about the world which is hidden from the state. An observation O is a partial description of a state, which may omit information. A history H is the sequence of observations, actions, rewards.

$$H_t = O_1, R_1, A_1, ..., A_{t-1}, O_t, R_t$$

State is a function of history.

$$S_t = f(H_t)$$

*Markov state property: A state S_t is Markov if and only if

$$P[S_{t+1}|S_t] = P[S_{t+1}|S_1, ..., S_t]$$

When the agent is able to observe the complete state of the environment, we say that the environment is fully observed. When the agent can only see a partial observation, we say that the environment is partially observed.
Fully observable:

$$O_t = S_t^e = St^a$$

Partially observable:

$$S_t^e \neq St^a$$

In this case, the agent must construct its own state representation.

2.2 Agent

2.2.1 Action

Different environments allow different kinds of actions. The set of all valid actions in a given environment is often called the **action space**.

Some environments have **discrete action spaces**, other environments have **continuous action spaces**. In continuous spaces, actions are real-valued vectors.

2.3 Reward

provided a **discounted rate** γ: $G_t = R_t + \gamma R_{t+1} + \gamma^2 R_{t+2} + ...$

2.4 Policy

A map from state to action. Can be deterministic or stochastic.

$$a = \mu(s) \quad or \quad a \sim \pi(a|s) = P[A = a|S = s]$$

2.5 Value function

A prediction of future reward

State value function

$$\begin{aligned} V_\pi(s_t) &= E_\pi[R_t|S_t = s_t] \\ &= \sum_{a \in \forall} \pi(a|s) q_\pi(s, a) \end{aligned}$$

Action-state value Function

$$\begin{aligned} Q_\pi(s_t, a_t) &= E_\pi[R_t | S_t = s_t, A_t = a_t] \\ &= \sum_{s^{'}, r} p(s^{'}, r | s, a)[r + \gamma * V_\pi(s^{'})] \end{aligned}$$

Bellman Expectation Equation

$$\begin{aligned} V_\pi(s) &= E_\pi[R_t + \gamma V_\pi(s_{t+1}) | S_t = s_t] \\ = \sum_{a \ \in \ \forall} \pi(a|s) &\sum_{s^{'}, r} p(s^{'}, r | s, a)[r + \gamma * V_\pi(s^{'})] \end{aligned}$$

$$\begin{aligned} Q_\pi(s_t, a_t) &= E_\pi[R_t + \gamma Q_\pi(s_{t+1}, a_{t+1}) | S_t = s_t, A_t = a_t] \\ &= \sum_{s^{'}, r} p(s^{'}, r | s, a)[r + \gamma * \sum_{a^{'} \in \forall} \pi(a^{'} | s^{'}) q_\pi(s^{'}, a^{'})] \end{aligned}$$

Optimal state value function

$$V^*(s) = \max_\pi V_\pi(s)$$

Optimal action-state value function

$$Q^*(s, a) = \max_\pi Q_\pi(s, a)$$

Bellman Optimal Equation

$$V^*(s) = \max_a Q^*(s, a)$$

$$Q^*(s, a) = \max_\pi Q_\pi(s, a) = \sum_{s^{'}, r} p(s^{'}, r | s, a)[r + \gamma * V^*(s^{'})]$$

2.6 Model

A model predicts what the environemt will do next.

$$P_{next\ state} = P\left[s_{t+1} | s_t, a_t\right]$$

$$R_{next\ reward} = R\left[r_{t+1} | s_t, a_t\right]$$

3 Model Free vs. Model Based

One of the most important branching points in an RL algorithm is the question of whether the agent has access to (or learns) a model of the environment. By a model of the environment, we mean a function which predicts state transitions and rewards.

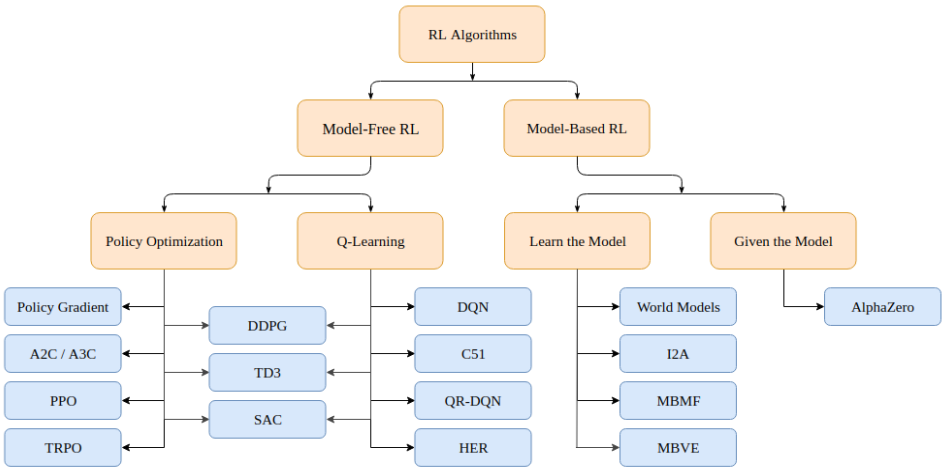

Figure 4: A Taxonomy of RL Algorithms

3.1 Planning vs. Learning

Two fundamental problems in sequential desicion making.

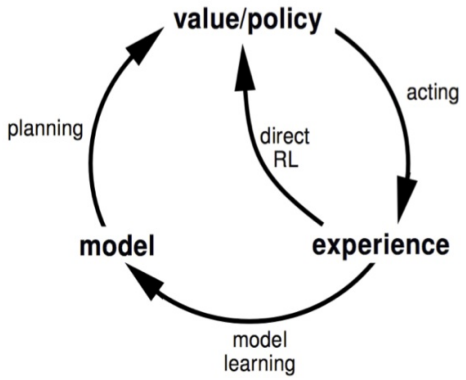

Figure 5: planning vs. learning

- Planning
  - rule of environment is known
  - agent only perform with its model
  - agent improves policy
- Learning
  - environment unknow

– agent directly interacts with environment

– agent imporves policy

3.2 Model Based

Model based algorithm allows the agent to plan. Consider the problem as given MDP (S.A.R.T), using **policy iteration** or **value iteration** to solve the problem. Most rely on DP.

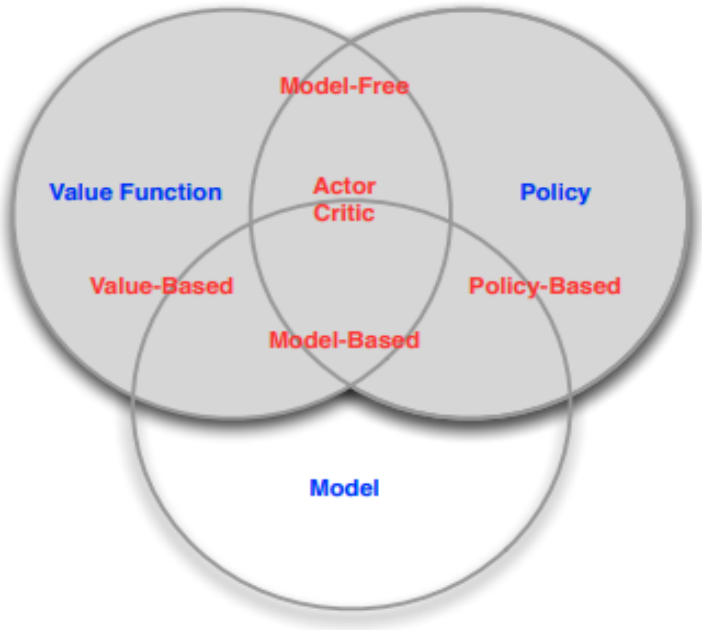

Figure 6: Model free vs. model based

3.3 Model Free

The agent does not know how the world will change in response to its actions (the transition function T), nor what immediate reward it will receive for doing so (the reward function R).

The agent will simply have to try taking actions in the environment, observe what happens, and somehow, find a good policy from doing so.

How? MC, TD...

3.3.1 Value-Based Reinforcement Learning

Suppose we already know $Q^*(s_t, a_t)$, It is easy to choose which action to make. DQN is one way that using a neural network $Q(s; a; w)$ to approximate $Q^*(s_t, a_t)$.

This optimization is almost always performed **off-policy**, which means that each update can use data collected at any point during training, regardless of how the agent was choosing to explore the environment when the data was obtained.

$$\begin{aligned} Q(s_t; a_t; w) &= r_t + \gamma \cdot Q(s_{t+1}; a_{t+1}; w) \\ &= r_t + \gamma \cdot \max_a Q(s_{t+1}; a; w) \end{aligned}$$

Off policy : Q-learning. On policy: SARSA

Initialize $Q(s, a)$ arbitrarily
Repeat (for each episode):
  Initialize S
  Repeat (for each step of episode):
    Choose A from S using policy derived from Q (e.g., ε-greedy)
    Take action A, observe R, S'
    $Q(S, A) \leftarrow Q(S, A) + \alpha[R + \gamma \max_a Q(S', a) - Q(S, A)]$
    $S \leftarrow S'$;
  until S is terminal

Figure 7: Q-learning

Initialize $Q(s, a)$ arbitrarily
Repeat (for each episode):
  Initialize S
  Choose A from S using policy derived from Q (e.g., ε-greedy)
  Repeat (for each step of episode):
    Take action A, observe R, S'
    Choose A' from S' using policy derived from Q (e.g., ε-greedy)
    $Q(S, A) \leftarrow Q(S, A) + \alpha[R + \gamma Q(S', A') - Q(S, A)]$
    $S \leftarrow S'$; $A \leftarrow A'$;
  until S is terminal

Figure 8: SARSA

3.3.2 Policy-Based Reinforcement Learning

Use a neural network $\pi(a|s; \theta)$ to approximate $\pi(a|s)$.

The goal of reinforcement learning is to find an optimal behavior strategy for the agent to obtain optimal rewards. The policy gradient methods target at modeling and optimizing the policy directly.

This optimization is almost always performed **on-policy**, which means that each update only uses data collected while acting according to the most recent version of the policy.

$$\begin{aligned} V(s_t;\theta) &= \sum_{a \in \forall} \pi(a|s_t;\theta) \cdot Q_\pi(s_t, a_t) \\ &= \int \pi(a|s_t;\theta) \cdot Q_\pi(s_t, a_t) da \end{aligned}$$

Policy gradient ascent to get max $J(\theta)$

$$J(\theta) = E_S[V(S;\theta)] = \sum_s V_\pi(s;\theta)$$

On Policy : REINFORCE, TRPO, PPO. Off Policy: DPG, Off-Policy Actor-Critic

3.3.3 Trade-offs between Policy-based and Value-based

Policy optimization methods

- directly optimize for the thing you want
- stable and reliable.

Value-based methods

- indirectly optimize for agent performance, by training Q to satisfy a self-consistency equation.
- tends to be less stable
- sample efficient, can reuse data more effectively than policy optimization