Reinforcement Learning

Summary

Fabian Damken

July 24, 2022

TECHNISCHE
UNIVERSITÄT
DARMSTADT

Contents

List of Figures

List of Tables

List of Algorithms

1 Introduction

In this course we will look at lots of methods from the domain of *reinforcement learning (RL)*. RL is an approach for agent-oriented learning where the agent learns by repeatedly acting with the environment and from rewards. Also, it does not know how the world works in advance. RL is therefore close to how humans learn and tries to tackle the fundamental challenge of artificial intelligence (AI):

"The fundamental challenge in artificial intelligence and machine learning is learning to make good decisions under uncertainty." (Emma Brunskill)

RL is so general that every AI problem can be phrased in its framework of learning by interacting. However, the typical setting is that at every time step, an agent perceives the state of the environment and chooses an action based on these perceptions. Subsequently, the agent gets a numerical reward and tries to maximize this reward by finding a suitable strategy. This procedure is illustrated in Figure 1.1.

1.1 Artificial Intelligence

The core question of AI is how to build "intelligent" machines, requiring that the machine is able to adapt to its environment and handle unstructured and unseen environments. Classically, AI was an "engine" producing answers to various queries based on rules designed by a human expert in the field. In (supervised) machine learning (ML), the rules are instead learned from a (big) data set and the "engine" produces answers based on the data. However, this approach (leaning from labeled data) is not sufficient for RL as demonstrations might be imperfect, the correspondence problem, and that we cannot demonstrate everything. We can break these issues down as follows: supervised learning does not allow "interventions" (trial-and-error) and evaluative feedback (reward).

The core idea leading to RL was to not program machines to simulate an adult brain, but to simulate a child's brain that is still learning. RL formalizes this idea of intelligence to interpret rich sensory input and choosing complex actions. We know that this may be possible as us humans do it all the time. This lead to the RL view on AI depicted in Figure 1.1 and is based on the hypothesis that learning from a scalar reward is sufficient to yield intelligent behavior (Sutton and Barto, 2018).

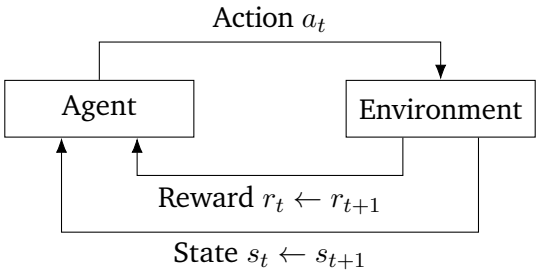

Figure 1.1: The Reinforcement Learning Cycle

	actions *do not* change the state of the world	actions change the state of the world
no model	(Multi-Armed) Bandits	Reinforcement Learning
known model	Decision Theory	Optimal Control, Planning

Table 1.1: Problem Classification

1.2 Reinforcement Learning Formulation

RL tries to *maximize the long-term reward* by finding a strategy/policy with the general assumption that it is easier to assess a behavior by specifying a cost than specifying the behavior directly. In general, we have the following things different to most (un)supervised settings:

- no supervision, but only a reward signal
- feedback (reward) is always delayed and not instantaneous
- time matters, the data is sequential and by no means i.i.d.
- the agent's actions influence the subsequent data, i.e., the agent generates its own data

In addition to this, RL is challenged by a numerous complicated factors and issues, e.g., dynamic state-dependent environments, stochastic and unknown dynamics and rewards, exploration vs. exploitation, delayed rewards (how to assign a temporal credit), and very complicated systems (large state spaces with unstructured dynamics). For designing an RL-application, we usually have to choose the state representation, decide how much prior knowledge we want to put into the agent, choose an algorithm for learning, design an objective function, and finally decide how we evaluate the resulting agent. By all these decisions, we want to reach a variety of goals, e.g., convergence, consistency, good generalization abilities, high learning speed (performance), safety, and stability. However, we are usually pretty restricted in terms of computation time, available data, restrictions in the way we act (e.g., safety constraints), and online vs. offline learning.

This sounds like a lot and, in fact, is! We therefore often limit ourselves onto specific (probably simpler) sub-problems and solve them efficiently under some assumptions. Some common flavors of the RL problem are, for instance:

- *Full:* no additional assumptions, the agent can only probe the environment through the state dynamics and its actions; the agent has to understand the environment
- *Filtered State and Sufficient Statistics:* assumption of a local Markov property (i.e., the next state only depends on the current state and action, and not on the past), decomposable rewards (into specific time steps); we can show that every problem is a (probably infinite) instance of this assumption, but how to filter the state to get such properties?
- *Markovian Observable State:* assume that we can observe the state fulfilling the Markov property directly
- *Further Simplifications:* contextual bandits (the dynamics do not depend on the action or the past and current state at all); bandits (only a single state)

We can summarize the different RL-like problems in a matrix, see Table 1.1.

1.2.1 Components

To solve an RL problem, we need three ingredients:

1. Model Learning
    - we want to approximate and learn the state transfer using methods from supervised learning
    - need to generate actions for model identification
    - estimation of the model or the model's parameters
2. Optimal Control/Planning
    - generation of optimal control inputs
3. Performance Evaluation

1.3 Wrap-Up

- why RL is crucial for AI and why all other approaches are ultimately doomed
- background and characteristics of RL
- classification of RL problems
- core components of RL algorithms

2 Preliminaries

2.1 Functional Analysis

2.1.1 Normed Vector Spaces

2.1.2 Contractions

2.1.3 Fixed Point (Theorem)

2.2 Statistics

2.2.1 Stochastic Processes

2.2.2 Monte-Carlo Estimation

2.2.3 Bias-Variance Trade-Off

2.2.4 Important Sampling

2.2.5 Linear Function Approximation

Feature Construction

Polynomial Features

Fourier Basis

Coarse Coding

Tile Coding

Radial Basis Functions

Neural Networks

2.2.6 Fisher Information Matrix

2.2.7 Entropy and Relative Entropy

2.2.8 Reparametrization Trick

2.3 Miscellaneous

2.3.1 Useful Integrals

2.3.2 Conjugate Gradient

3 Markov Decision Processes and Policies

3.1 Markov Decision Processes

3.1.1 Continuous State-Action-Space

3.1.2 Example

3.2 Markov Reward Processes

3.2.1 Return and Discount

3.2.2 Value Function

Bellman Equation

3.2.3 Example

3.3 Markov Decision Processes

3.3.1 Policies

Value Functions

Bellman Expectation Equation

Bellman Operator

Optimality

Bellman Optimality Equation

Bellman Optimality Operator

3.3.2 Example

3.4 Wrap-Up

4 Dynamic Programming

4.1 Finite Horizon DP

4.2 Policy Iteration

4.2.1 Policy Evaluation

4.2.2 Policy Improvement

4.2.3 Using the Action-Value Function

4.2.4 Examples

4.3 Value Iteration

4.3.1 Principle of Optimality

4.3.2 Convergence

4.3.3 Example

4.4 Policy vs. Value Iteration

4.5 Efficiency

5 Monte-Carlo Algorithms

5.1 Policy Evaluation

5.2 Example

6 Temporal Difference Learning

6.1 Temporal Differences vs. Monte-Carlo

6.1.1 Bias-Variance Trade-Off

6.1.2 Markov Property

6.1.3 Backup

6.2 Bootstrapping and Sampling

6.3 TD(λ)

6.3.1 n-Step Return

6.3.2 λ-Return

6.3.3 Eligibility Traces

6.4 Example

6.5 Wrap-Up

7 Tabular Reinforcement Learning

7.0.1 Monte-Carlo Methods

Generalized Policy Iteration

Greediness and Exploration vs. Exploitation

ϵ-Greedy Exploration and Policy Improvement

Monte-Carlo Policy Iteration and Control

GLIE Monte-Carlo Control

7.0.2 TD-Learning: SARSA

Convergence

n-Step

Eligibility Traces and SARSA(λ)

Example

7.1 Off-Policy Methods

7.1.1 Monte-Carlo

7.1.2 TD-Learning

Importance Sampling

Q-Learning

Convergence

Example

7.2 Remarks

7.3 Wrap-Up

8 Function Approximation

8.1 On-Policy Methods

8.1.1 Stochastic Gradient Descent

8.1.2 Gradient Monte-Carlo

. . . with Linear Function Approximation

8.1.3 Semi-Gradient Methods

. . . with Linear Function Approximation

8.1.4 Least-Squares TD

Semi-Gradient SARSA

8.2 Off-Policy Methods

8.2.1 Semi-Gradient TD

8.2.2 Divergence

8.3 The Deadly Triad

8.4 Offline Methods

8.4.1 Batch Reinforcement Learning

8.4.2 Least-Squares Policy Iteration

8.4.3 Fitted Q-Iteration

8.5 Wrap-Up

9 Policy Search

9.1 Policy Gradient

9.1.1 Computing the Gradient

Finite Differences

Least-Squares-Based Finite Differences

Likelihood-Ratio Trick

9.1.2 REINFORCE

Gradient Variance and Baselines

Example

9.1.3 GPOMDP

9.2 Natural Policy Gradient

9.3 The Policy Gradient Theorem

9.3.1 Actor-Critic

9.3.2 Compatible Function Approximation

Example

9.3.3 Advantage Function

9.3.4 Episodic Actor-Critic

9.4 Wrap-Up

10 Deep Reinforcement Learning

10.1 Deep Q-Learning: DQN

10.1.1 Replay Buffer

10.1.2 Target Network

10.1.3 Minibatch Updates

10.1.4 Reward- and Target-Clipping

10.1.5 Examples

10.2 DQN Enhancements

10.2.1 Overestimation and Double Deep Q-Learning

10.2.2 Prioritized Replay Buffer

10.2.3 Dueling DQN

10.2.4 Noisy DQN

10.2.5 Distributional DQN

10.2.6 Rainbow

10.3 Other DQN-Bases Methods

10.3.1 Count-Based Exploration

10.3.2 Curiosity-Driven Exploration

10.3.3 Ensemble-Driven Exploration

10.4 Wrap-Up

11 Deep Actor-Critic

11.1 Surrogate Loss

11.1.1 Kakade-Langford-Lemma

11.1.2 Practical Surrogate Loss

11.2 Advantage Actor-Critic (A2C)

11.3 On-Policy Methods

11.3.1 Trust-Region Policy Optimization (TRPO)

Practical Implementation

11.3.2 Proximal Policy Optimization (PPO)

11.4 Off-Policy Methods

11.4.1 Deep Deterministic Policy Gradient (DDPG)

11.4.2 Twin Delayed DDPG (TD3)

11.4.3 Soft Actor-Critic (SAC)

11.5 Wrap-Up

12 Frontiers

12.1 Partial Observability

12.2 Hierarchical Control

12.2.1 The Options Framework

12.3 Markov Decision Processed Without Reward

12.3.1 Intrinsic Motivation

12.3.2 Inverse Reinforcement Learning

12.4 Model-Based Reinforcement Learning

12.5 Wrap-Up