Active Learning for Reward Modelling

Sam Clarke

September 2019

Abstract

Contents

Chapter 1

Introduction

1.1 Relation to Material Studied on the MSc Course

Chapter 2

Reinforcement Learning

Reinforcement learning (RL) refers simultaneously to a problem, methods for solving that problem, and the field that studies the problem and its solution methods. The problem of RL is to learn what to do—how to map situations to actions—so as to maximise some numerical reward signal [3, pp. 1-2].

In this section we first introduce the elements of RL informally. We then formalise the RL Problem as the optimal control of incompletely-known Markov decision processes (finite? do I talk about PO?). We give a taxonomy of different RL solution methods and conclude with a description of one such method, Deep Q-Learning (DQN) that is of particular importance in this dissertation.

2.1 Elements of Reinforcement Learning

This subsection will introduce agent, environment, policy, reward signal, value function and [model] informally, similar to S&B 1.3. Is this necessary or should I skip straight to the formalism?

2.2 Finite Markov Decision Processes

Finite Markov Decision Processes (finite MDPs) are a way of mathematically formalising the RL problem: they capture the most important aspects of the problem

faced by an agent interacting with its environment to achieve a goal. We introduce the elements of this formalism: the agent-environment interface, goals and rewards, returns and episodes. Then...

2.2.1 The Agent-Environment Interface

MDPs consist firstly of the continual interaction between an agent selecting actions, and an environment responding by changing state, and presenting the new state to the agent, along with an associated scalar reward. Recall that the agent seeks to maximise this reward over time through its choice of actions.

More formally, consider a sequence of discrete time steps, $t = 1, 2, 3, \ldots$. At each time step t, the agent receives some representation of the environment's *state*, $S_t \in \mathcal{S}$, and chooses an *action*, $A_t \in \mathcal{A}$. On the next time step, the agent receives reward $R_{t+1} \in \mathcal{R} \subset \mathbb{R}$, and finds itself in a new state, S_{t+1}. These interactions repeat over time, giving rise to a *trajectory*:

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

A *finite* MDP is one where the sets of states, actions and rewards are finite. In this case, the random variables S_t and R_t have well-defined discrete probability distributions which depend only on the preceding state and action. This allows us to define the *dynamics* of the MDP, a probability mass function $p : \mathcal{S} \times \mathcal{R} \times \mathcal{S} \times \mathcal{A} \mapsto [0, 1]$, as follows. For any particular values $s' \in \mathcal{S}$ and $r \in \mathcal{R}$ of the random variables S_t and R_t, there is a probability of these values occurring at time t, given any values of the previous state $s \in \mathcal{S}$ and action $a \in \mathcal{A}$:

$$p(s', r \mid s, a) := \mathbb{P}\left(S_t = s', R_t = r \mid S_{t-1} = s, A_{t-1} = a\right).$$

A *Markov* Decision Process is one where all states satisfy the Markov property.

A state s_t satisfies this property iff:

$$\mathbb{P}\left(s_{t+1} \mid s_t, s_{t-1}, s_{t-2}, \ldots, s_t\right) = \mathbb{P}\left(s_t\right).$$

This implies that in the MDP, the immediately preceding state s_t and action a_t are sufficient statistics for predicting the next state s_{t+1} and reward r_{t+1}.

2.2.2 Goals and Rewards

The reader may have noticed that we first introduced MDPs as a formalism for an agent interacting with its environment to achieve a goal, yet have since spoken instead of maximising a reward signal $R_t \in \mathbb{R}$ over time. Our implicit assumption is the following hypothesis:

Hypothesis 1 (Reward Hypothesis)**.** *All of what we mean by goals and purposes can be well thought of as the maximization of the expected value of the cumulative sum of a received scalar signal (called reward). [3, p. 53]*

However, this hypothesis gives no information about how to construct such a scalar signal; only that it exists. Indeed, recent work has shown that it is far from trivial to do so; possible failure modes include negative side effects, reward hacking and unsafe exploration [1]. This is central to the topic of this dissertation—our aim is to improve the sample efficiency of one particular method of reinforcement learning when the reward signal is unknown.

2.2.3 Returns and Episodes

Having asserted that we can express the objective of reinforcement learning in terms of scalar reward, we now formally define this objective. Consider the following objective:

Definition 1 ((Future discounted) return)**.** *Let a sequence of rewards between time step $t+1$ and T (inclusive) be $R_{t+1}, R_{t+1}, \ldots, R_T$. Let $\gamma \in [0, 1]$ be a discount factor*

of future rewards. Then we define the (future discounted) return of this sequence of rewards [3, p. 57],

$$G_t := \sum_{k=t+1}^{T} \gamma^{k-t-1} R_k. \tag{2.1}$$

This definition is convenient because it allows us to describe two kinds of reinforcement learning tasks, episodic and continuing, with the same notation. In episodic tasks, interactions between the agent and environment occur in well-defined subsequences, each of which ends in a special *terminal state*. The environment then resets to a starting state, which may be fixed or sampled from a distribution. In continuing tasks, on the other hand, agent-environment interaction does not naturally divide into episodes.

For episodic tasks, it makes sense to consider (2.1) in case T is the time step at which the episode ends, because we typically analyse either a single episode or something that holds across all episodes [3, p. 57]. We call this *episode* return. In this case, (2.1) is well-defined regardless of the discount factor γ. For continuous tasks, $T = \infty$ thus (2.1) may be infinite. However, if $\gamma < 1$ then the infinite sum (2.1) will be finite in more cases. For example, if reward is a constant $+r$, then the return is:

$$G_t = \sum_{k=0}^{\infty} \gamma^k r = \frac{r}{1-\gamma}, \tag{2.2}$$

whereas the return would have been infinite if we did not use discounting.

2.2.4 Policies and Value Functions

Policy determines the behaviour of the agent. Formally, a policy $\pi : \mathcal{S} \times \mathcal{A} \mapsto [0, 1]$ defines a probability distribution over actions, given a state. That is to say, $\pi(a \mid s)$ is the probability of selecting action a if an agent is following policy π and in state s.

The *state-value function* $v_\pi : \mathcal{S} \mapsto \mathbb{R}$ for a policy π gives the expected return of starting in a state and following that policy. More formally,

Definition 2 (State-value function)**.** *Let π be a policy and $s \in \mathcal{S}$ be any state. We*

write $\mathbb{E}_\pi[.]$ *to denote the expected value of the random variable* G_t *as defined in (2.1). Then,*

$$v_\pi(s) := \mathbb{E}_\pi[G_t \mid S_t = s] = \mathbb{E}_\pi\left[\sum_{k=0}^{\infty} \gamma^k R_{t+k+1} \mid S_t = s\right]. \tag{2.3}$$

The *action-value function* $q_\pi : \mathcal{S} \times \mathcal{A} \mapsto \mathbb{R}$ for a policy π is defined similarly. It gives the expected return of starting in a state, taking a given action, and following policy π thereafter.

Definition 3 (Action-value function)**.** *Let* π *be a policy,* $s \in \mathcal{S}$ *be any state and* $a \in \mathcal{A}$ *any action. The action-value function*

$$q_\pi(s,a) := \mathbb{E}_\pi[G_t \mid S_t = s, A_t = a] = \mathbb{E}_\pi\left[\sum_{k=0}^{\infty} \gamma^k R_{t+k+1} \mid S_t = s, A_t = a\right]. \tag{2.4}$$

2.2.5 Optimal Policies and Optimal Value Functions

2.2.6 Reinforcement Learning Solution Methods

Reinforcement learning methods specify how an agent should use its experience update its policy π.

Deep Q-Learning

2.3 Reinforcement Learning from Unknown Reward Functions

For many domains in which we might want to use RL, states of the environment are not inherently associated with rewards. Thus, we have the additional task of specifying a reward function, which maps states to rewards. Argue the case that directly specifying a reward function is hard, in order to motivate the next section: Reward Learning

2.3.1 Reward Learning

In Standard RL

In the Deep Case

Chapter 3

Uncertainty in Deep Learning

Standard deep learning models output point estimates. For example, a model trained to classify pictures of dogs according to their breed takes a picture of a dog and outputs its predicted breed. However, what will the model do if it is given a picture of a cat? [2].

We probably want the model to be able to recognise that this is an out of distribution example, and request more training data, or simply say that it doesn't know the answer. However, since standard deep learning models output only point estimates, the model will just go ahead and classify the cat as some breed of dog, just as confidently as any other input.

Thus, the field of Bayesian Deep Learning aims to equip neural networks with the ability to output a point estimate along with its uncertainty in that estimate. Historically, many of the attempts to do so were not very practical. For example, one algorithm, Bayes by Backprop, requires doubling the number of model parameters, making training more computationally expensive, and is very sensitive to hyperparameter tuning. However, recent techniques allow almost any network trained with a stochastic regularisation technique, such as dropout, to, given an input, obtain a predictive mean and variance (uncertainty), without any complicated augmentation to the network [2, p. 15].

Chapter 4

Active Learning

4.0.1 Applying Active Learning to RL without a reward function

Chapter 5

Method

Chapter 6

Experiments

Chapter 7

Results

Chapter 8

Conclusions

8.1 Summary

8.2 Evaluation

8.3 Future Work

References

[1] Dario Amodei, Chris Olah, Jacob Steinhardt, Paul Christiano, John Schulman, and Dan Mané. Concrete Problems in AI Safety. pages 1–29, 2016.

[2] Yarin Gal. Uncertainty in Deep Learning. *Phd Thesis*, 1(1):1–11, 2017.

[3] R. S. Sutton and A. G. Barto. *Reinforcement Learning: An Introduction (2nd Edition, in preparation)*. 2018.

Appendices

Appendix A

Some Appendix Material