Abstract

abstract in German in at least 500 words is a requirement!!!!

optionally also in English

write this last with introduction

Contents

1 Introduction 1

2 Background on Formal Languages 2

3 Background on Reinforcement Learning 3

4 Reward Machines 9

5 Conclusion & Outlook 10

References 11

A Appendix 12

1 Introduction

write this last with abstract

2 Background on Formal Languages

write after two other chapters (15-20 pages)

preliminaries to cover:

- regular languages (LTL, regular expressions)
- (deterministic) finite state automata (DFA)
- mealy/moore machines
- büchli automata

3 Background on Reinforcement Learning

(10-15 pages)

cover:

- MDP
- policy
- q function
- Bellman equations
- q learning

The goal of reinforcement learning is to learn a task by maximizing the total rewards along sequences of decisions called episodes. The difficulty is that decisions can have delayed consequences. TODO: explain this more naturally

We have an agent that acts in an environment. How the environment behaves is unknown to the agent. Furthermore for each action the agent takes it receives a reward from the environment, but how these rewards are determined is also not visible to the agent. So the environment is a complete black box to the agent.

Consider the following example as a demonstration of the reinforcement learning framework. In a grid-based office environment a robot is tasked with bringing coffee from the kitchen to a particular office, see figure 1 for the layout.

> maybe a better example would be a grid world where the agent should try to climb the hill and not fall into the pit or something along those lines, see [1]

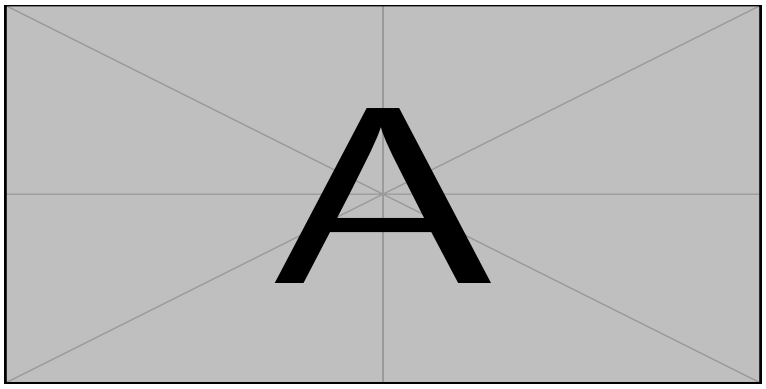

Figure 1: office world... TODO!

The environment starts in state S_0 and for each state $S_t \in \mathcal{S}$ at time $t \in \mathbb{N}$ the agent has to decide on an action $A_t \in \mathcal{A}$ like going up, down, left, right or use the coffee machine. After executing the action the environment changes to a new state S_{t+1} and rewards the

robot with a reward $R_{t+1} \in \mathcal{R} \subseteq \mathbb{R}$. This feedback loop is shown as a diagram in figure 2.

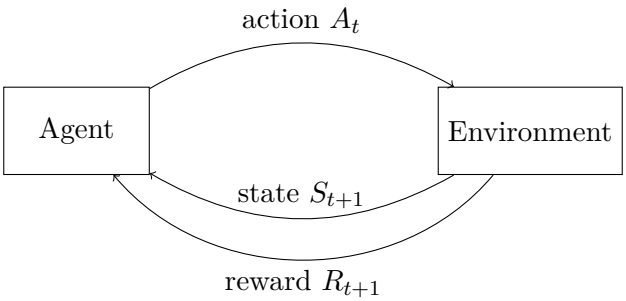

Figure 2: Diagram of the reinforcement learning feedback loop.

From the point of view of the agent the environment and the rewards are random processes. Here for example $(S_t)_{t\in\mathbb{N}} \subseteq \mathcal{S}$ describes the evolution of the states and $(R_t)_{t\in\mathbb{N}} \subseteq \mathcal{R}$ the rewards, which are defined on some probability space with joint measure $\mathbb{P}$. A key assumption is that these processes do not remember any history other than the state that they are currently in, i.e. $\mathbb{P}(S_{t+1} = s, R_{t+1} = r \mid S_0, A_0, S_1, R_1, A_1, \ldots, S_t, R_t, A_t) = \mathbb{P}(S_{t+1} = s, R_{t+1} = r \mid S_t, A_t)$. This means that we assume that the environment has the Markov property and altough in real life scenarios this might not be quite true it is still a good approximation and the theory for non-Markovian processes also uses the ideas we will develop in this chapter.

Following [1] we will also assume finite spaces for the sake of clarity but the theory can also be developed using probability densities on infinite spaces. Usually we do not care about the details of these processes and underlying probability spaces, but we work with some derived quantities instead, namely the state-transition probability distribution $p(s'|s,a) = \sum_{r\in\mathcal{R}} \mathbb{P}(S_{t+1} = s', R_{t+1} = r \mid S_t = s, A_t = a)$ and the expected reward $r(s,a) = \mathbb{E}(R_{t+1} \mid S_t = s, A_t = a) = \sum_{s'\in\mathcal{S}} \sum_{r\in\mathcal{R}} \mathbb{P}(S_{t+1} = s', R_{t+1} = r \mid S_t = s, A_t = a)$. This motivates the defining model of RL, the Markov decision process: TODO: motivate more why we want markov property

Definition 3.1. *A* Markov Decision Process (MDP) *is given by a tuple* $(\mathcal{S}, \mathcal{A}, p, r, \gamma)$*, where* $\mathcal{S}$ *is a finite set of* states*,* A *is a finite set of* actions*,* $p : S \times S \times A \to [0,1]$ *is the* state-transition probability distribution*,* $r : S \times A \to \mathbb{R}$ *is the* reward function *and* $\gamma \in [0,1]$ *is a* discount factor*. Note that here implicitly* $\mathcal{R}$ *is the image of* r *and thus a finite subset of* $\mathbb{R}$*.*

This MDP models the diagram in figure 2 from above as follows: From a given state $s \in \mathcal{S}$ the agent chooses an action $a \in \mathcal{A}$ and the environment changes to state $s' \in \mathcal{S}$ with probability $p(s'|s,a)$ and gives reward $r(s,a)$. The action space can actually depend on the state the agent is currently in, like in our example the agent cannot go outside the boundary, so the action space is limited when the agent is at the wall. But since we assume finite states, this distinction does not matter, so we will omit it for simplicity.

The discount factor will become relevant in a moment but in essence a lower discount factor would motivate the agent to take actions based on the reward sooner rather than later as with a higher discount factor.

In the office world example ... TODO: what is the mdp in the example

TODO: motivate a policy

Definition 3.2. *A* (deterministic) policy $\pi : \mathcal{S} \to \mathcal{A}$ *is a function that prescribes an action* $\pi(s) \in \mathcal{A}$ *to a given state* $s \in \mathcal{S}$.

A (non-deterministic) policy $\pi : \mathcal{A} \times \mathcal{S} \to [0, 1]$ *is a probabiliy distribution that describes the probability* $\pi(a|s)$ *of taking an action* $a \in \mathcal{A}$ *in the given state* $s \in \mathcal{S}$.

In the following we will freely alternate between the two, whatever is most convenient. Whenever we write $\pi(s)$ we mean a deterministic policy and when we write $\pi(a|s)$ we mean a non-deterministic one.

For our office world example a policy could be to go left when in room 1 and go right when in room 2 or stand still when not in either room. Of course this would not be a good policy, but what constitutes a "good" policy will be defined with the state value function and the action value function.

Definition 3.3 (Value of Policies)**.** *(i) The* state-value function*, often just* value function*,* $v_\pi : \mathcal{S} \to \mathbb{R}$ *under a policy* π *is defined as the expected discounted sum of future rewards starting from a state* $s \in \mathcal{S}$ *and taking actions according to* π*:*

$$v_\pi(s) = r_\pi(s) + \gamma \sum_{s' \in \mathcal{S}} \left(p_\pi(s'|s) r_\pi(s') + \gamma \sum_{s'' \in \mathcal{S}} \left(p_\pi(s''|s') r_\pi(s'') + \dots \right) \right)$$

where $p_\pi(s'|s) = \sum_{a \in \mathcal{A}} \pi(a|s) p(s'|s, a)$ *and* $r_\pi(s) = \sum_{a \in \mathcal{A}} \pi(a|s) r(s, a)$*. Since we only consider finite spaces in this setting we get the following in vector form:*

$$\begin{aligned} V_\pi &= R_\pi + \gamma \sum_{s' \in \mathcal{S}} \left((P_\pi)_{s,s'} (R_\pi)_s + \gamma \sum_{s'' \in \mathcal{S}} \left((P_\pi)_{s',s''} (R_\pi)_{s''} + \dots \right) \right) \\ &= R_\pi + \gamma \sum_{s' \in \mathcal{S}} (P_\pi)_{s,s'} (R_\pi)_{s'} + \gamma^2 \sum_{s' \in \mathcal{S}} (P_\pi^2)_{s,s'} (R_\pi)_{s'} + \dots \\ &= R_\pi + \gamma P_\pi V_\pi \end{aligned}$$

where $P_\pi \in [0, 1]^{|\mathcal{S}| \times |\mathcal{S}|}$ *is the transition matrix,* $R_\pi \in \mathbb{R}^{|\mathcal{S}|}$ *the vector of all rewards and* $V_\pi \in \mathbb{R}^{|\mathcal{S}|}$ *the state-values vector.*

(ii) The action-value function *or* Q-function $q_\pi : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$ *under a policy* π *is the expected discounted sum of taking action* $a \in \mathcal{A}$ *in state* $s \in \mathcal{S}$ *and then following policy* π*:*

$$q_\pi(s, a) = r(s, a) + \gamma \sum_{s' \in \mathcal{S}} p(s'|s, a) v_\pi(s')$$

Now that we have defined the value of a policy for each state we can define when a policy is considered "better" than another.

Definition 3.4. *We say $\pi \geq \pi'$ if $v_\pi(s) \geq v_{\pi'}(s)$ for all states $s \in \mathcal{S}$, which naturally defines a partial order on the space of policies.*

If we already have a policy we can improve upon it by utilising its q-function. TODO: expand upon this

Proposition 3.5 (Greedy Policy Improvement)**.** *For a policy π define a deterministic policy $\pi'(s) = \arg\max_{a \in \mathcal{A}} q_\pi(s, a)\ \forall s \in \mathcal{S}$. Then $\pi' \geq \pi$.*

Proof. TODO! □

This can be used as an algorithm to find a good policy by iteratively improving the policy. One issue with this greedy policy improvement is that not all state-action pairs are visited. It will always select the immediate best option and not explore other options, which is why it is called greedy. The issues that this can have will be more clear later in this chapter. But basically the algorithm could get stuck in a local optimum for a while or if we only have an approximation to the environment, it may be subotimal. We can help with this problem by turning the improved policy into a non-deterministic one. Instead of always selecting the greedy action we can select it with probability $1 - \varepsilon$ and all other actions with equal likelihood.

Proposition 3.6 (ε-Greedy Policy Improvement)**.** *For a policy π define the ε-greedy policy for $0 < \varepsilon \ll 1$ and $\forall s \in \mathcal{S}$.*

$$\pi'(a|s) = \begin{cases} 1 - \varepsilon + \frac{\varepsilon}{|\mathcal{A}|} & \text{if } a = \arg\max_{a' \in \mathcal{A}} q_\pi(s, a') \\ \frac{\varepsilon}{|\mathcal{A}|} & \text{else} \end{cases}$$

Then $\pi' \geq \pi$.

Proof. TODO: or just write analagously? □

Naturally, the question arises if there is a best policy and if the (ε-)greedy policy improvement converges. The answer to both is yes.

Proposition 3.7 (Existence of Optimal Policy)**.** *For a finite MDP there exists an optimal policy.*

Proof. TODO! □

Iteratively improving the policy using the above schemes until the value functions no longer increase, we get an algorithm that converges to an optimal policy. This optimal policy need not be unique though.

Proposition 3.8 ((ε-)Greedy Policy Improvement Convergence)**.** *Using (ε-)greedy policy improvement, when $v_{\pi'} = v_\pi$, then $\pi' = \pi = \pi_*$ is an optimal policy.*

Proof. TODO! □

In order to use this algorithm though, we need to determine $\arg\max_{a\in\mathcal{A}} q_\pi(s,a)$. But to calculate $q_\pi(s,a)$ we would need to know the transition probabilites $p(s'|s.a)$ of the environment as well as the rewards $r(s,a)$. For non-toy examples we usually do not know these, so we would need a way to model the environment. Alternatively we can approach this without a model and use so called model-free algorithms. To construct a model-free algorithm, we will use a property that an optimal policy π_* satisfies, called the Bellman optimality equation.

Proposition 3.9 (Bellman Optimality Equation)**.** *If π^* is an optimal policy, then its value functions $v_* = v_{\pi^*}$ and $q_* = q_{\pi^*}$ satisfy the following.*

$$v_*(s) = \max_{a\in\mathcal{A}} r(s,a) + \gamma \sum_{s'\in\mathcal{S}} p(s'|s,a) v_*(s')$$

$$q_*(s,a) = r(s,a) + \gamma \sum_{s'\in\mathcal{S}} p(s'|s,a) \max_{a'\in\mathcal{A}} q_*(s',a')$$

Proof. TODO! □

- use simple example (mountain car? crossing river?)
- existence of optimal policy
- bellman optimality equation
- value iteration
- notes on model/model-free and DP
-
- model-free is using q, using v requires model
- first q learning or DP? https://www.wikiwand.com/en/Dynamic_programming, i.e. using recursion to break down into simpler subproblems and then can use techniques such as memoization
- DP requires complete knowledge, but even with its computationally too expensive
- use order from lecture:
  1. MRP
  2. value function

3. Bellman equations for MRP
4. MDP
5. Policy
6. action value function
7. optimal policy
8. Bellman optimality equations
9. Bellman expectation operator
10. policy evaluation
11. policy improvement
12. policy improvement convergence
13. Bellman optimality operator
14. value iteration
15. notes on DP: not model-free / computationally expensive
16. episode
17. first visit monte carlo
18. temporal difference
19. greedy policy improvement
20. exporation and ε-greedy
21. GLIE
22. GLIE Monte carlo
23. On-Policy SARSA
24. On/Off-Policy
25. Off-Policy Q-Learning

4 Reward Machines

(15-20 pages)

papers:

- original reward machine paper from 2018 [2]
- connection to LTL from 2019 [3]
- newer reward machine paper from 2022 [4]

include updated RL framework graph to better differentiate that reward machines are not in the environment black box

include same task from previous chapter and how a reward machine for that might look like as a graph

central theorems:

- construction/correspondence to LTL and such
- convergence proof

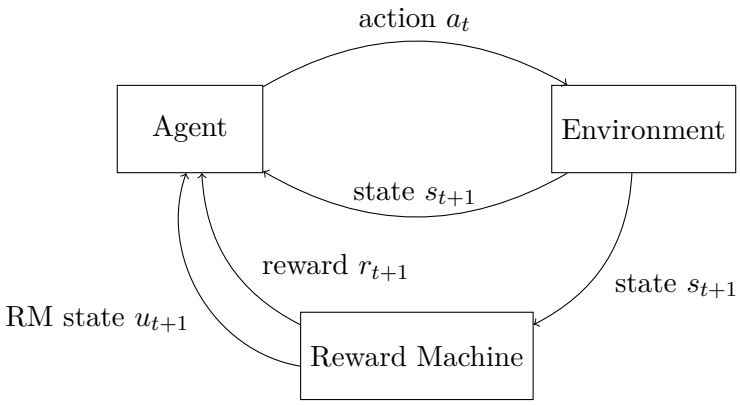

Figure 3: Diagram of the reinforcement learning feedback loop with a reward machine. The agent now not only gets an environment state but also a reward machine state.

The mathematical model for this is ...

Definition 4.1. *A* Markov Decision Process with Reward Machine (MDPRM) *is ...*

5 Conclusion & Outlook

write this after the main chapters

References

[1] R.S. Sutton and A.G. Barto. *Reinforcement Learning, second edition: An Introduction*. Adaptive Computation and Machine Learning series. MIT Press, 2018. ISBN: 9780262039246. URL: https://books.google.at/books?id=5s-MEAAAQBAJ.

[2] Rodrigo Toro Icarte et al. "Using Reward Machines for High-Level Task Specification and Decomposition in Reinforcement Learning". In: *Proceedings of the 35th International Conference on Machine Learning (ICML)*. 2018, pp. 2112–2121. URL: http://proceedings.mlr.press/v80/icarte18a.html.

[3] Alberto Camacho et al. "LTL and Beyond: Formal Languages for Reward Function Specification in Reinforcement Learning". In: *Proceedings of the 28th International Joint Conference on Artificial Intelligence (IJCAI)*. 2019, pp. 6065–6073. URL: https://www.ijcai.org/proceedings/2019/0840.pdf.

[4] Rodrigo Toro Icarte et al. "Reward Machines: Exploiting Reward Function Structure in Reinforcement Learning". In: *Journal of Artificial Intelligence Research (JAIR)* 73 (2022), pp. 173–208. URL: https://doi.org/10.1613/jair.1.12440.

A Appendix

experiments and code go here convergence graphs of qrm/crm and standard