SAPIENZA
UNIVERSITÀ DI ROMA

Reward shaping in RL for $\mathrm{LTL}_f/\mathrm{LDL}_f$ Goals: Theory and Practice

Facoltà di Ingegneria dell'Informazione, Informatica e Statistica

Corso di Laurea Magistrale in Master in Engineering in Computer Science

Candidate

Marrco Favorito

ID number 1609890

Thesis Advisor

Prof. Giuseppe De Giacomo

Co-Advisor

Dr. co-advisor

Academic Year 2017/2018

Reward shaping in RL for $\mathrm{LTL}_f/\mathrm{LDL}_f$ Goals: Theory and Practice
Master thesis. Sapienza – University of Rome

This thesis has been typeset by LaTeX and the Sapthesis class.

Author's email: favorito.1609890@studenti.uniroma1.it

Abstract

write your abstract here

Contents

Chapter 1

Introduction

Chapter 2

Preliminaries

In this chapter we describe the background knowledge required for this work. We introduce Markov Decision Process (MDP) and Non-Markovian Reward Decision Process (NMRDP), common formalisms in the context of Reinforcement Learning. We describe Linear Temporal Logic over finite traces (LTL$_f$) and Linear Dynamic Logic over finite traces (LDL$_f$), that we use for define temporal goal in a RL setting. Then, we describe an important result about RL for NMRDP with LTL$_f$/LDL$_f$ rewards, that is the basis for this work.

2.1 Reinforcement Learning

Reinforcement Learning (Sutton and Barto, 1998) is a sort of optimization problem where an *agent* interacts with an *environment* and obtains a *reward* for each action he chooses and the new observed state. The task is to maximize a numerical reward signal obtained after each action during the interaction with the environment. The agent does not know a priori how the environment works (i.e. the effects of his actions), but he can make observations in order to know the new state and the reward. Hence, learning is made in a *trial-and-error* fashion. Moreover, it is worth to notice that in many situation reward might not been affected only from the last action but from an indefinite number of previous action. In other words, the reward can be *delayed*, i.e. the agent should be able to foresee the effect of his actions in terms of future expected reward.

In the next subsections we introduce some of the classical mathematical frameworks for RL: Markov Decision Process (MDP) and Non-Markovian Reward Decision Process (NMRDP).

2.1.1 Markov Decision Process

A Markov Decision Process (MDP) $\mathcal{M}$ is a tuple $\langle S, A, T, R, \gamma \rangle$ containing a set of *states* S, a set of *actions* A, a *transition function* $T : S \times A \to Prob(S)$ that returns for every pair state-action a probability distribution over the states, a *reward function* $R : S \times A \times S \to \mathbb{R}$ that returns the reward received by the agent when he performs action a in s and transitions in s', and a *discount factor* γ, with $0 \leq \gamma \leq 1$, that indicates the present value of future rewards.

A *policy* $\rho : S \to A$ for an MDP $\mathcal{M}$ is a mapping from states to actions, and represents a solution for $\mathcal{M}$. Given a sequence of rewards $R_{t+1}, R_{t+2}, \ldots, R_T$, the *expected return* G_t at time step t is defined as:

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

where can be $T = \infty$ and $\gamma = 1$ (but not both).

The *value function* of a state s, the *state-value function* $v_\rho(s)$ is defined as the expected return when starting in s and following policy ρ, i.e.:

$$v_\rho(s) \coloneqq \mathbb{E}_\rho[G_t|S_t = s], \forall s \in S \tag{2.2}$$

Similarly, we define q_ρ, the *action-value function for policy* ρ, as:

$$q_\rho(s,a) \coloneqq \mathbb{E}_\rho[G_t|S_t = s, A_t = a], , \forall s \in S, \forall a \in A \tag{2.3}$$

Notice that we can rewrite 2.2 and 2.3 recursively, yielding the *Bellman equations*:

$$v_\rho(s) = \sum_{s'} P(s'|s,a)[R(s,a,s') + \gamma v(s')] \tag{2.4}$$

where we used the definition of the transition function:

$$T(s,a,s') = P(s'|s,a) \tag{2.5}$$

We define the *optimal state-value function* and the *optimal action-value function* as follows:

$$v^*(s) \coloneqq \max_\rho v_\rho(s), \forall s \in S \tag{2.6}$$

$$q^*(s,a) \coloneqq \max_\rho q_\rho(s,a), \forall s \in S, \forall a \in A \tag{2.7}$$

Notice that with 2.6 and 2.7 we can show the correlation between $v^*_\rho(s)$ and $q^*_\rho(s,a)$:

$$q^*(s,a) = \mathbb{E}_\rho[R_{t+1} + \gamma v^*_\rho(S_{t+1})|S_t = s, A_t = a] \tag{2.8}$$

We can define a partial order over policies using value functions, i.e. $\forall s \in S . \rho \geq \rho' \iff v_\rho(s) \geq v_{\rho'}(s)$. An *optimal policy* ρ^* is a policy such that $\rho^* \geq \rho$ for all ρ.

2.1.2 Temporal Difference Learning

Temporal difference learning (TD) refers to a class of model-free reinforcement learning methods which learn by bootstrapping from the current estimate of the value function. These methods sample from the environment, like Monte Carlo (MC) methods, and perform updates based on current estimates, like dynamic programming methods (DP). We do not discuss MC and DP methods here.

Q-Learning and Sarsa are such a methods. They updates $Q(s,a)$, i.e. the estimation of $q^*(s,a)$ at each transition $(s,a) \to (s',r)$. Q-Learning uses the following update rule:

$$Q(s,a) \leftarrow Q(s,a) + \alpha[r + \gamma \max_{a'} Q(s',a') - Q(s,a)] \tag{2.9}$$

while Sarsa:

$$Q(s,a) \leftarrow Q(s,a) + \alpha[r + \gamma Q(s',a') - Q(s,a)] \tag{2.10}$$

TD(λ) is an algorithm which uses *eligibility traces*. The parameter λ refers to the use of an eligibility trace. The algorithm generalizes MC methods and TD learning, obtained respectively by setting $\lambda = 1$ and $\lambda = 0$. Intermediate values of λ yield methods that are often better of the extreme methods. Q-Learning and Sarsa that has been shown before can be rephrased with this new formalism as Q-Learning(0) and Sarsa(0), special cases of Watkin's Q(λ) and Sarsa(λ).

2.2 LTL$_f$ and LDL$_f$

In this section we introduce LTL$_f$ and LDL$_f$, two formalisms that we will use for define the reward function of a RL task over *sequence of transitions* rather than one transition.

2.2.1 Linear Temporal Logic on Finite Traces: LTL$_f$

Linear-time Temporal Logic over finite traces, LTL$_f$, is essentially standard LTL (Pnueli, 1977) interpreted over finite, instead of over infinite, traces (De Giacomo and Vardi, 2013).

Indeed, the syntax of LTL$_f$ is the same of LTL$_f$, i.e. *formulas* of LTL$_f$ are built from a set $\mathcal{P}$ of propositional symbols and are closed under the boolean connectives, the unary temporal operator $\circ$(*next-time*) and the binary operator $\mathcal{U}$ (*until*):

$$\varphi \quad ::= \quad \phi \mid \neg\varphi \mid \varphi_1 \wedge \varphi_2 \mid \circ\varphi \mid \varphi_1 \mathcal{U} \varphi_2$$

With $A \in \mathcal{P}$.

We use the standard abbreviations: $\varphi_1 \vee \varphi_2 \doteq \neg(\neg\varphi_1 \wedge \neg\varphi_2)$; *eventually* as $\Diamond\varphi \doteq true\,\mathcal{U}\,\varphi$; *always* as $\Box\varphi \doteq \neg\Diamond\neg\varphi$; week next $\bullet\varphi \doteq \neg\circ\neg\varphi$ (note that on finite traces $\neg\circ\varphi \not\equiv \circ\neg\varphi$); and *Last* $\doteq \bullet$*false* denoting the end of the trace.

Formally, a *finite trace* π is a finite word over the alphabet $2^{\mathcal{P}}$, i.e. as alphabet we have all the possible propositional interpretations of the propositional symbols in $\mathcal{P}$. For the semantics we refer to (De Giacomo and Vardi, 2013).

LTL$_f$ is as expressive as first-order logic (FO) over finite traces and star-free regular expressions (RE).

2.2.2 Linear Dynamic Logic on Finite Traces: LDL$_f$

LTL$_f$ can be extended to LDL$_f$, *Linear Dynamic Logic of Finite Traces*

which is expressive as monadic second-order logic (MSO) over finite traces De Giacomo and Vardi (2013).

Linear Dynamic Logic of Finite Traces

2.2.3 LTL$_f$ and LDL$_f$ translation to automata

2.3 RL for NMRDP with LTL$_f$/LDL$_f$ rewards

In this section we introduce the formalism of Non-Markovian Reward Decision Process (?)

2.3.1 Non-Markovian Reward Decision Process

A Non-Markovian Reward Decision Process (NMRDP) $\mathcal{N}$ is a tuple $\langle S, A, T, \bar{R}, \gamma \rangle$ where everything is defined as in the MDP but the reward function is defined as $\bar{R} : (S \times A)^* \to \mathbb{R}$, i.e. is defined over sequences of states and actions. Given a trace $\pi = \langle s_0, a_0, s_1, a_1, \ldots, s_n, a_n \rangle$, the *value of* π is:

$$v(\pi) = \sum_{i=1}^{|\pi|} \gamma^{i-1} \bar{R}(\langle \pi(1), \pi(2), \ldots, \pi(i) \rangle)$$

where $\pi(i) = (s_i, a_i)$.

The policy $\bar{\rho}$ in this setting is defined over sequences of states, i.e. $\bar{\rho} : S^* \to A$. The *value of* $\bar{\rho}$ given an initial state s_0 is defined as:

$$v^{\bar{\rho}}(s) = \mathbb{E}_{\pi \sim \mathcal{M}, \bar{\rho}, s_0}[v(\pi)]$$

i.e. the expected value in state s considering the distribution of traces defined by the transition function of $\mathcal{M}$, the policy $\bar{\rho}$ and the initial state s_0.

Chapter 3

RL for $\text{LTL}_f/\text{LDL}_f$ Goals

3.1 RL for NMRDP with $\text{LTL}_f/\text{LDL}_f$ rewards

3.2 RL for $\text{LTL}_f/\text{LDL}_f$ Goals

Chapter 4

Automata-based Reward shaping

4.1 Reward Shaping

4.2 Reward shaping over automaton φ

Chapter 5

FLLOAT

Chapter 6

RLTG

Chapter 7

Experiments

7.1 BREAKOUT

7.2 SAPIENTINO

7.3 MINECRAFT

Chapter 8

Conclusions

Bibliography

Giuseppe De Giacomo and Moshe Y Vardi. Linear temporal logic and linear dynamic logic on finite traces. In *IJCAI*, volume 13, pages 854–860, 2013.

Amir Pnueli. The temporal logic of programs. In *Proceedings of the 18th Annual Symposium on Foundations of Computer Science*, SFCS '77, pages 46–57, Washington, DC, USA, 1977. IEEE Computer Society. doi: 10.1109/SFCS.1977.32. URL https://doi.org/10.1109/SFCS.1977.32.

Richard S. Sutton and Andrew G. Barto. *Introduction to Reinforcement Learning*. MIT Press, Cambridge, MA, USA, 1st edition, 1998. ISBN 0262193981.