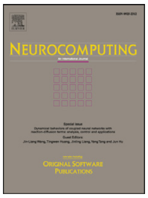
Neurocomputing 000 (2017) 1–9

Contents lists available at ScienceDirect

Neurocomputing

journal homepage: www.elsevier.com/locate/neucom



Identification and off-policy learning of multiple objectives using adaptive clustering

Thommen George Karimpanal*, Erik Wilhelm

Engineering Product Development, Singapore University of Technology and Design, 8 Somapah Road, Singapore 487372, Singapore

ARTICLE INFO

Article history:
Received 4 March 2016
Revised 1 March 2017
Accepted 21 April 2017
Available online xxx

Keywords:
Reinforcement learning
Q-learning
Off-policy
Adaptive clustering
Multiobjective learning

ABSTRACT

In this work, we present a methodology that enables an agent to make efficient use of its exploratory actions by autonomously identifying possible objectives in its environment and learning them in parallel. The identification of objectives is achieved using an online and unsupervised adaptive clustering algorithm. The identified objectives are learned (at least partially) in parallel using $Q-$learning. Using a simulated agent and environment, it is shown that the converged or partially converged value function weights resulting from off-policy learning can be used to accumulate knowledge about multiple objectives without any additional exploration. We claim that the proposed approach could be useful in scenarios where the objectives are initially unknown or in real world scenarios where exploration is typically a time and energy intensive process. The implications and possible extensions of this work are also briefly discussed.

1. Introduction

Intelligent agents are characterized by their abilities to learn from and adapt to their environments with the objective of performing specific tasks. Very often, in reinforcement learning [1], and in machine learning in general, algorithms are structured to be able to fulfill one specific objective, usually specified in terms of a particular region in the feature space that is associated with a high reward. In general, environments are likely to contain multiple features, and different regions in the feature space may specify different objectives that could be assigned to the agent to learn. In real-world scenarios, however, the ability to efficiently learn more than one objective during a single deployment could drastically improve the agent's usefulness. In order to achieve this, the agent would need to be aware of regions in the feature space that could possibly play a role in its future tasks.

Embodied artificial agents or intelligent robots are typically equipped with a variety of sensors that enable them to detect characteristic features in their environment. In the context of reinforcement learning, when such an agent is placed in an unknown environment and is assigned an objective, it carries out some form of exploratory behavior in order to first discover a region in the feature space that fulfills this objective. Further exploratory actions may help improve its value function estimates, which in turn lead to improved policies to achieve the objective. We shall refer to this original task as the *primary objective*, and to its associated feature vector as the *primary objective feature vector* ($\vec{\psi}$). During exploration, it is likely that the agent comes across other 'interesting' regions which contain features that stand out with respect to the agent's history of experiences. We shall refer to these regions of the feature space as *secondary objectives*, and to the associated feature vectors as *secondary objective feature vectors* ($\vec{\phi}$). Although these regions could be of interest to the agent for future tasks (which are currently unknown), they may be irrelevant to the task at hand. Hence, it is justified for the agent to ignore them and continue performing value function updates for the primary objective assigned to it.

However, the agent's future tasks may not remain the same and a new task assigned to it may correspond to a particular combination of features that it encountered while learning policies for the primary objective. In such a case, the fact that this region in the feature space had been previously encountered cannot be leveraged since they were not relevant to the agent at that point of time, and were hence ignored.

The above mentioned approach would result in a considerable amount of wasteful exploration. This is because each new task assigned to the agent would require a fresh phase of discovery and learning of the associated feature vector and value functions, respectively. A more efficient approach would be to keep track of possible secondary objectives and learn them in parallel using off-policy methods [1,2]. In the context of off-policy learning, this can

* Corresponding author.
E-mail addresses: thommen_george@mymail.sutd.edu.sg, thommengk@gmail.com (T.G. Karimpanal), erikwilhelm@sutd.edu.sg (E. Wilhelm).

http://dx.doi.org/10.1016/j.neucom.2017.04.074
0925-2312/© 2017 Elsevier B.V. All rights reserved.

be done by treating the policies corresponding to the secondary objectives as target policies, and learning them while executing the behavior policy which is dictated by the primary objective. Depending on the objectives, the actions executed by the behavior policy may not be optimal with respect to the secondary objectives. However, using off-policy learning, it is possible to at least partially learn the value functions for the secondary objectives, thereby significantly improving the efficiency of exploration. In applications such as robotics where exploration is known to be costly in terms of time, energy and other factors, such an approach could prove to be practical.

In this work, we present a framework in which an unsupervised, adaptive clustering algorithm is designed and used to cluster regions of the feature space into different groups based on the similarity of their associated features. Off-policy methods are used to simultaneously learn target policies corresponding to these clusters, each of which is treated as a secondary objective. The clustering of features occurs as and when they are seen by the agent while learning the primary task. The value function updates can be performed using suitable off-policy methods, namely, tabular $Q-$ learning, $Q-\lambda$ [3] or other more recent off-policy methods [4] such as off-policy LSTD(λ) [5,6], off-policy TD(λ) [2,7], GQ(λ) [8] etc., The results presented here, however, correspond to the $Q-\lambda$ algorithm.

The primary objectives have an influence on the discovery and learning of the secondary objectives, but only through its behavior policy. As long as the agent executes some exploratory actions while learning to perform its primary task, secondary objectives can be discovered and at least partially be learned. In fact, even a purely exploratory policy can be used. These aspects are discussed in further detail in Section 5.

Ideally, our approach would obviate the need for a fresh phase of discovery and learning when the objective is changed. However, the aim here is not to learn all the secondary objectives perfectly, but to identify them via the adaptive clustering algorithm, and learn them at least partially through off-policy learning. Doing so could provide the agent with a good initialization of value function weights so that optimal policies for the identified possible objectives could be learned in the future, if needed.

2. Background

Reinforcement learning deals with developing strategies for an agent to act in its environment with the objective of maximizing the expected value of a scalar reward. Most research in reinforcement learning is based on the formalism of Markov Decision Processes (MDPs) [9]. In this framework, an agent in state $s \in \mathcal{S}$ takes an action $a \in \mathcal{A}$ to transition into a new state s' with a probability $P(s, a, s')$. At each state, the agent receives a scalar reward $R(s, a)$. All reinforcement learning methods can be thought of as ways to maximize the expected reward accumulated over time as the agent interacts with the environment. The outcome of these methods is a mapping from states to actions, referred to as a policy. If the learning agent learns the value function for the policy being executed, it is referred to as *on-policy* learning, and if it learns the value function for an objective irrespective of the policy being executed, it is called *off-policy* learning.

In this work, our goal is to identify secondary objectives and learn their corresponding policies in parallel while the agent executes its behavior policy based on its primary objective. Hence, *off-policy* learning methods are a natural choice for the stated goal. We use the $Q-\lambda$ algorithm, which is an extension of tabular $Q-$learning that is suitable for application in continuous state spaces. The update equation for the tabular case is shown in Eq. (1)

$$Q(s,a) \leftarrow Q(s,a) + \alpha[R(s,a) + \gamma max_{a'} Q(s',a') - Q(s,a)] \tag{1}$$

where $Q(s, a)$ is the $Q-$value corresponding to state s and action a. s' is the next state, and a' is a bound variable that can represent any action in the action space $\mathcal{A}$. α is the learning rate and γ is the discount factor.

The $Q-\lambda$ algorithm performs a similar, but more involved update with weight vectors, and involves the use of eligibility traces [10]. Here, replacing traces are used for the $Q-\lambda$ updates [11]. The update equations for the $Q-\lambda$ algorithm are mentioned below:

$$\delta \leftarrow \delta + \gamma max_{a'} Q(s',a') \tag{2}$$

$$w \leftarrow w + \alpha\delta e \tag{3}$$

$$e \leftarrow \gamma\lambda e \tag{4}$$

where w is the weight vector, e is the eligibility trace vector, λ is the trace decay rate parameter and δ is defined as

$$\delta = R(s,a) - Q(s,a) \tag{5}$$

The elements of the eligibility trace vector (replacing traces) are initialized with a value of 1 if the corresponding features are active. Otherwise, they are initialized with a value of 0.

The $Q-$values mentioned in Eqs. (2) and (5) are stored in the form of weight vectors as:

$$Q(s,a) = \sum_{i \in \mathcal{F}_{act}(s,a)} w_i \tag{6}$$

where $F_{act}(s, a)$ is the set of active features for an agent in state s, taking an action a. A more detailed summary of the algorithm can be found in [1].

Although off-policy methods such as the ones described above have been well known and widely used over the years, their use for autonomously handling multiple independent objectives has been limited, primarily owing to very few precedents on unsupervised identification of objectives in an agent's environment. Off-policy approaches with function approximation have also been known to have long standing issues with stability until recently [12]. Although approaches for handling multiple independent objectives in parallel are rather limited, a number of multi-objective reinforcement learning approaches that handle multiple conflicting objectives exist. A comprehensive survey of such methods can be found in [13].

The horde architecture of Sutton et al. [12] has been shown to be able to learn multiple pre-defined objectives in parallel using independent reinforcement learning agents in an off-policy manner. The knowledge of these tasks is stored in the form of generalized value functions which makes it possible to obtain predictive knowledge relating to different goals of the agent. Modayil et al. [14] and White et al. [15] also focus on learning multiple objectives in parallel using off-policy learning. Apart from this, Sutton et al. [16] used off-policy methods to simultaneously learn multiple options [17], including ones not executed by the agent. They mention that the motivation for using off-policy methods is to make maximum use of whatever experience occurs and to learn as much as possible from them, which is an idea that is reflected in this work.

In the works mentioned above, the multiple objectives that are learned in parallel are pre-defined. However, in this work, we focus on the case where the agent has no foreknowledge of the objectives in its environment. The objectives are identified by the agent itself via clustering. Hence, the agent learns independently in the sense that as it moves through its environment, it identifies potential objectives and at least partially learns their associated value functions in parallel.

A similar approach is seen in Mannor et al. [18], where clustering is performed on the state-space to identify interesting regions. However, their approach was not online and the purpose of their