Review: MODRL

Norio Kosaka

December 16, 2018

1 Paper Profile

- Title: Multi-Objective Deep Reinforcement Learning
- Authors: Hossam Mossalam, Yannis M. Assael, Diederik M. Roijers, Shimon Whiteson
- Organisation: University of Oxford, CS Dept.
- Publish Year: Oct 2016
- URL: https://arxiv.org/pdf/1610.02707.pdf

2 Contents

3 Proposal

In this paper, they have proposed two things, which are vanilla DOL(Deep Optimistic Linear Support Learning) and standard benchmarks in MORL(Multi-Objective Reinforcement Learning) domain. First one is related to solve high-dimensional multi-objective decision problems where relative importance of the objectives are not known in advance. Also, since this is allegedly the first paper regarding MODRL ever, they have demonstrated their approaches on the novel benchmarks.

4 Introduction

- Recent advances in RL has been focusing on Single Objective Learning. [1-16]
- Since in general it is not clear how to evaluate available trade-offs among objectives in advance, it is appropriate to construct a **coverage set(CS)** which contains at least one optimal policy for each possible utility function.
- They have found that the reason why DL was not applicable for Markov Decision Processes(MDPs). One reason is that it is not clear how neural networks account for unknown preferences and the resulting sets of value vectors.
- Making use of Outer Loop Approach[19], they achieved to learn an approximate coverage set of policies , which is represented by a neural network, by evaluating a sequence of scalarised single-objective problems.
- Outer loop approach is that it repeatedly calls a single-objective problems and finds an optimal policy followed by the termination after finite number of calls to that subroutine. Then it produces an approximate CS.
- 3 propositions
  - OLS and its compliant
  - Deep OLS Learning(DOL)
  - DOL with Full/Partial Reuse

5 Background

This section covers mainly four topics,

- Markov Decision Process(MDP)
- Deep Q-networks(DQN)
- Multi-objective MDPs(MOMDP)
- Optimistic Linear Support(OLS)

Assuming the readers' understanding on the basic of RL, I would like to briefly state the formulae for each MDP and DQN.

- MDP: $< S, A, R, T, \gamma >$, where S: state space, A: action space, R: Reward function, γ: discount factor for the future uncertainty
- DQN: $L_i(\theta_i) = E_{s,a,r,s'}[(r + \gamma max_{a'} Q(s', a'; \theta_i^-) - Q(s, a, ; \theta_i))]$

Multi-Objective MDPs: An MOMDP is an variant of MDP in which the reward function is represented by a vector of n rewards, one for each objective. Also, the solution to these problem is a set of policies called a *coverage set* containing at least one optimal policy for each possible preference; utility or scalarisation function f, that a user might have. It maps each possible policy value vector, V^π onto a scalar value. In this paper, they have focused on the case where the scalarisatoin function is linear $f(V^\pi, w) = w \cdot V^\pi$, where w is a vector that determines the relative importance of the objectives, and this function itself is called a *convex coverage set*, for more detail I would like to refer readers to the nice power point created by Diederik M. Roijers(http://roijers.info/pub/vubSep14.pdf) which formally defined the *convex coverage set*.

Optimistic Linear Support(OLS): OSL takes an *outer loop approach* in which the CCS is incrementally constructed by solving a series of scalarised MDPs for different linear scalarisation vectors w. And this allows us to adapt DQNs to s single-objective MDP problem decomposed from the main MOMDP problem. In contrast to the *outer loop approach*, OLS utilises the concept of *corner weights* to pick the weights to use for creating scalarised instances and the concept of estimated improvement to prioritise those corner weights. Intuitively describing, it is the method to find the upper bound which arises in the area above all crossed utility lines on Figure1. According to the paper, those corners have to be initialised heuristically in practice.

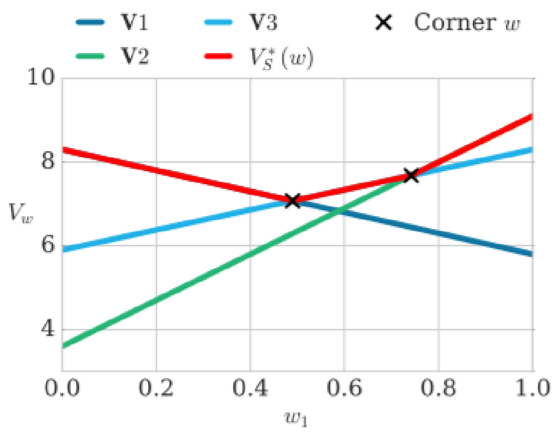

Figure 1: The two corner weights of a $V_S^*(\mathbf{w})$ with S containing three value vectors for a 2-objective MOMDP.

6 Methdology

In this section, the authors have written about their two propositions as following.

- Deep OLS Learning, which is described in Algorithm 1.
- Deep OLS Learning with Full(DOL-FR) and Partial Reuse(DOL-PR)

6.1 Deep OLS Learning

There are two requirements to adapt DQN on the OLS framework.

1. OLS compliant scalarised function
2. Modification of the output of DQN to obtain a matrix of action values for each objective.

These requirement led them to the novel approach which is a scalarised DQN to comprise further development.

Algorithm 1 Deep OLS Learning (with different types of reuse)

```
function DOL($m, \tau$, template, reuse)
    ▷ Where, $m$ – the (MOMDP) environment, $\tau$ – improvement threshold,
    ▷ template – specification of DQN architecture, reuse – the type of reuse
    $S$ = empty partial CSS
    $W$ = empty list of explored corner weights
    $Q$ = priority queue initialised with the extrema weights simplex with infinite priority
    DQN_Models = empty table of DQNs, indexed by the weight, $\mathbf{w}$, for which it was learn
    while $Q$ is not empty $\wedge$it $\leq$ max_it do
        $\mathbf{w} = Q$.pop()
        if reuse = 'none' $\vee$ DQN_Models is empty then
            model = a randomly initialised DQN, from a pre-specified architecture templat
        else
            model = copyNearestModel($\mathbf{w}$, DQN_Models)
            if reuse = 'partial' then reinitialise the last layer of model with random weights
        $\mathbf{V}$, new_model = scalarisedDeepQLearning($m$, $\mathbf{w}$, model)
        $W = W \cup \mathbf{w}$
        if $(\exists \mathbf{w}')\ \mathbf{w}' \cdot \mathbf{V} > \max_{\mathbf{U} \in S} \mathbf{w}' \cdot \mathbf{U}$ then
            $W_{del} = W_{del} \cup$ corner weights made obsolete by $\mathbf{V}$ from $Q$
            $W_{del} = W_{del} \cup \{\mathbf{w}\}$
            Remove $W_{del}$ from $Q$
            Remove vectors from $S$ that are no longer optimal for any $\mathbf{w}$ after adding $\mathbf{V}$
            $W_{\mathbf{V}}$ = newCornerWeights(S, $\mathbf{V}$)
            $S = S \cup \{\mathbf{V}\}$
            DQN_Models[$\mathbf{w}$] = new_model
            for each $\mathbf{w}' \in W_{\mathbf{V}}$ do
                if estimateImprovement($\mathbf{w}', W, S$) > $\tau$ then
                    $Q$.add($\mathbf{w}'$)
        it ++
    return $S$, DQN_Models
```

6.2 Deep OLS Learning with Full(DOL-FR) and Partial Reuse(DOL-PR)

They found the issue in relearning the parameters for the entire network when it moves on to the next single-objective problem using the OLS framework. I would like to leave the details of the algorithm to the original paper though, I will quickly describe main characteristic of two approaches.

- DOL-FR: we start learning for a new scalarisation weight w', using the complete network we optimised for the previous w.
- DOL-PR: we take the same network as full-reuse, but we re-initialise the last layer randomly to avoid the local optima.

7 Experimental Evaluation

They have verified the performance of their approaches usign two game environments which is *Mountain Car(MC)* and *Deep Sea treasure(DST)* with corresponding modification in the input of raw images. But no comparison to other existing approaches.

8 Related Work

In this section, they mentioned two recently progressing domains.

- Inner Loop Approach: not applicable to DQN
- Evolutionary Algorithms: slow convergence compared to back-propagation...