Multi-Objective Reinforcement Learning using Sets of Pareto Dominated Policies

Hasan Mert Gökalp

December 12, 2022

Overview

Introduction

- Many real-life problems involve dealing with multiple objectives.
- No single optimal solution
- We want to obtain the best trade-off solutions(pareto front)
- Single-policy algorithms and multi-policy algorithms
- Reinforcement learning for multi-objective problems.

Markov Decision Process A Markov decision process is a 4-tuple (S, A, T_a, R_a) where:

- $S = (s_1, ..., s_N)$ the state space
- $A = (a_1, ..., a_r)$ the action set
- $T(s'|s, a)$ the transition probability
- R(s,a) the expected immediate reward

Policies

- The goal is to learn a deterministic stationary policy π
- The state-dependent value function:

$$V^{\pi}(s) = E_{\pi}[\sum_{k=0}^{\infty} \gamma^{k} r_{t+k+1} | s_t = s]$$

- The action value function: $Q^{\pi}(s, a)$ stores the expected return starting from state s, taking action a, and thereafter following π again

Q-Learning

- The optimal Q^*-values:

$$Q^*(s,a) = R(s,a) + \gamma \sum_{s'} T(s' \mid s,a) \max_{a'} Q^*(s',a)$$

- Each entry contains a value for $\hat{Q}(s,a)$ which is the learner's current estimate about the actual value of $Q^*(s,a)$

$$\hat{Q}(s,a) \leftarrow (1-\alpha_t)\,\hat{Q}(s,a) + \alpha_t \left(r + \gamma \max_{a'} \hat{Q}\left(s',a'\right)\right)$$

Reinforcement Learning

Algorithm 1 Single-objective Q-learning algorithm

Initialize $\hat{Q}(s, a)$ arbitrarily
for each episode t **do**
    Initialize s
    repeat
        Choose a from s using a policy
        Take action a and observe s'
        $\hat{Q}(s, a) \leftarrow (1 - \alpha_t)\,\hat{Q}(s, a) + \alpha_t \left(r + \gamma \max_{a'} \hat{Q}\,(s', a')\right)$
        $s \leftarrow s'$
    until s is terminal
end for

Multi Objective

- In the case of MORL, MDP provide a vector of rewards

$$R(s,a) = (R_1(s,a), ...R_m(s,a))$$

- the state-dependent value function of a state s is vectorial

$$V^{\pi}(s) = E_{\pi}[\sum_{k=0}^{\infty} \gamma^k r_{t+k+1} | s_t = s]$$

- Since the environment now consists of multiple objectives, different policies can be optimal w.r.t. different objectives

Pareto front

Definition

A policy π_1 is said to strictly dominate another solution π_2, that is $\pi_2 \prec \pi_1$, if each objective in V^{π_1} is not strictly less than the corresponding objective of V^{π_2} and at least one objective is strictly greater.A policy π is Pareto optimal if V^{π} either strictly dominates or is incomparable with the value functions of the other policies. The set of Pareto optimal policies is referred to as the Pareto front.

Single-policy Algorithms

Most approaches of reinforcement learning on multi-objective tasks rely on single-policy algorithms

- employ scalarization functions

$$v_w = f(v, w)$$

- scalar $\hat{Q}$-values are extended to $\hat{Q}$ -vectors

$$\hat{Q}(s, a) = (\hat{Q}_1(s, a), ...\hat{Q}_m(s, a))$$

- scalarization of $\hat{Q}$ vector for using traditional strategies:

$$\hat{SQ}(s, a) \leftarrow f(\hat{Q}(s, a), w)$$

Scalarization Algorithms

Selecting an action in a certain state of the environment

Algorithm 2 Scalarized ϵ-greedy strategy

```
SQList ← {}
for each action a in A do
    v ← Q̂_1(s, a), ..., Q̂_m(s, a)
    SQ̂(s, a) ← f(v, w)
    Append SQ̂(s, a) to SQList
end for
return ϵ-greedy(SQList)
```

Algorithm 3 Scalarized multi-objective Q-learning algorithm

Initialize $\hat{Q}_o(s, a)$ arbitrarily
for each episode t **do**
    Initialize state s
    repeat
        Choose action a from s using the policy derived from $\hat{SQ}$-values
        Take action a and observe s'
        $a' \leftarrow greedy(s')$
        for each objective o **do**
            $\hat{Q}_o(s, a) \leftarrow (1 - \alpha_t)\, \hat{Q}_o(s, a) + \alpha_t \left(r + \gamma \hat{Q}_o\, (s', a') \right)$
        end for
        $s \leftarrow s'$
    until s is terminal
end for

Multi-Policy MORL

- In contrast to single-policy MORL, multi-policy algorithms do not reduce the dimensionality of the objective space but aim to learn a set of optimal solutions at once
- dynamic programming (DP) function:

$$\hat{Q}_{set}(s, a) = \mathbf{R}(s, a) \oplus \gamma \sum_{s' \in S} T\left(s' \mid s, a\right) V^{ND}\left(s'\right)$$

- The ND operator is a function that removes all Pareto dominated elements of the input set and returns the set of non-dominated elements

$$V^{ND}\left(s'\right) = ND\left(\cup_{a'} \hat{Q}_{set}\left(s', a'\right)\right)$$

Pareto Q-Learning

- Assumption:Problem is episodic
- It does not assume any model
- 3 set evaluation mechanisms
- The set-based bootstrapping problem(lack of correspondence)
- Learning Immediate And Future Reward Separately
- $\bar{\Re}(s, a)$:= average observed immediate reward vector of (s, a)
  $ND_t(s, a)$:= set of non-dominated vectors in the next state of s that is reached through action a at time step t.

$$\hat{Q}_{set}(s, a) \leftarrow \bar{\Re}(s, a) \oplus \gamma ND_t(s, a)$$

Algorithm 4 Pareto Q-learning algorithm

Initialize $\hat{Q}_{set}(s, a)$'s as empty sets
for each episode t **do**
  Initialize state s
  repeat
    Choose action a using a policy derived from the $\hat{Q}_{set}(s, a)$'s
    Take action a and observe state s' and reward vector r
    $ND_t(s, a) \leftarrow ND\left(\cup_{a'} \hat{Q}_{set}\left(s', a'\right)\right)$
    $\bar{\mathfrak{R}}(s, a) \leftarrow \bar{\mathfrak{R}}(s, a) + \frac{\mathbf{r} - \bar{\mathfrak{R}}(s,a)}{n(s,a)}$
    $s \leftarrow s'$
  until s is terminal
end for

Set Evaluation Mechanisms

- The hypervolume measure is a quality indicator that evaluates a particular set of vectorial solutions by calculating the volume with respect to its elements and a reference point
- The only quality indicator to be strictly monotonic with the Pareto dominance relation

Algorithm 5 Hypervolume Q_{set}evaluation

```
Retrieve current state s
evaluations = { }
for each action a do
    hv_a ← HV(Q̂_set(s, a))
    Append hv_a to evaluations
end for
return evaluations
```

Set Evaluation Mechanisms

- consider the number of Pareto dominating $\hat{Q}$-vectors of the $\hat{Q}$-set of each action
- the action that relates to the largest number of Pareto dominating Qˆ -vectors over all actions in s is selected.

Algorithm 6 Cardinality Q_{set} evaluation

```
Retrieve current state s
allQs = { }
for each action a in s do
    for each Q̂ in Q̂_set(s, a) do
        Append [a, Q̂] to allQs
    end for
end for NDQs ← ND(allQs)
return NDQs
```

Tracking a Policy

- one needs to select actions consistently in order to retrieve a desired policy

---

Algorithm 7 Track policy given the expected reward vector

---

target $\leftarrow V^{\pi}(s)$
repeat
    for each a in A **do**
        Retrieve $\bar{\Re}(s, a)$
        Retrive $ND_t(s, a)$
        for each Q in $ND_t(s, a)$ **do**
            if $\gamma Q + \bar{\Re}(s, a) =$ target **then**
                $s \leftarrow s' \; : \; T(s'|s, a) = 1$
                $target \leftarrow Q$
            end if
        end for
    end for
until s is not terminal