Sorting Permutations by Reversals and Transpositions with Reinforcement Learning

Guilherme Bueno Andrade *Andre Rodrigues Oliveira*
Zanoni Dias

Relatório Técnico - IC-PFG-18-13
Projeto Final de Graduação
2018 - Julho

UNIVERSIDADE ESTADUAL DE CAMPINAS
INSTITUTO DE COMPUTAÇÃO

Sorting Permutations by Reversals and Transpositions with Reinforcement Learning

Guilherme Bueno Andrade* Andre Rodrigues Oliveira† Zanoni Dias†

Abstract

TODO

1 Introduction

TODO

2 Reinforcement Learning

2.1 Overview

Reinforcement Learning (RL), like other branches of Machine Learning, has been drawing a lot of attention from the community in the recent years. Google DeepMind's AlphaGo victory over Lee Sedol [1] - world champion of the game Go, is one of many examples of recent astonishing applications of the technique. It consists of an agent learning how to accomplish a certain goal based on interactions with the environment.

Initially, the agent receives a state $S0$. Based on that, the agent then takes an action $A0$, ending up at state $S1$ and receiving some reward $R1$. This process keeps going until the agent reaches a terminal state. Its goal is to maximize the total reward it gets along the way; i.e., $\max \sum_t R_t$.

In order for the agent to accomplish such task, in value-based RL - the one being considered in this work, it optimizes the value function $V(s)$,

$$V(s) = \mathbb{E}[R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+3} + \ldots | S_t = s] \tag{1}$$

where $\gamma \in [0, 1)$ is a discount rate that makes the agent care more about most likely short term reward, and less about less probable future rewards.

*gbuenoandrade@gmail.com

†Institute of Computing, University of Campinas, Brazil.

2.2 Exploitation vs. Exploration

A major concern in RL is the exploitation/explorarion trade-off. Exploration is about exploring new possibilities within the environment and finding out more information about it. Exploitation, on the other hand, is related to exploiting already known information so as to maximize the total reward.

Initially, the agent has no other option but to randomly explore the environment; however, as it learns about its surroundings, it can fall into the trap of sticking to safe known actions and miss larger rewards that depend on exploring unknown states.

This work uses the Epsilon-greedy strategy to address that problem. It specifies an exploration rate ϵ, which is set to 1 initially. This rate definies the ratio of the steps that will be done randomly. Before selecting an action, the agent generates a random number x. If $x > \epsilon$, then it will select the best known action (exploit); otherwise, it will select an action at random (explore). As the agent acquires more knowledge about the environment, ϵ is progressively reduced.

2.3 Q-table and the Bellman Equation

From the value function defined in equation 1, we have the Q-table, defined as follows.

$$Q(s,a) = \mathbb{E}[R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+3} + \ldots | S_t = s, A_t = a] \tag{2}$$

Where s' is the state that will be reached after the agent performs action a.

This is convenient because it allows the agent to pick the best action that can be performed from state s by simply finding $\arg\max_a Q(s,a)$.

Furthermore, Q can be expressed in terms of itself. An expression known as the *Bellman Equation* (ref).

$$Q(s,a) = \mathbb{E}[R_{t+1} + \gamma \sum_{a'} Q(s', a')] \tag{3}$$

The above form is handy because it opens doors for iterative approaches such as dynamic programming (ref).

2.4 Function Approximation

If one can calculate the Q-table from equation 3, they can successfully build an agent that maximizes the total reward. As mentioned in the last section, this can be easily done with dynamic programming. However, as the number of states grows largers, dynamic programming or other iterative approaches becomes unfeasible due to memory and time constraints. Fortunately, it turns out that the Q-table can be approximated instead of having its exact values determined, and it still produces great results. This work tries to achieve that using linear regression and deep neural networks.

2.5 TD(0) and Monte Carlo Method

TODO

2.6 TD-Lambda

TODO

2.7 Deep Q-learning

TODO

3 Experiment

3.1 Modeling

TODO

3.2 Results and Discussion

TODO

References

[1] T. Guardian. Alphago seals 41 victory over go grandmaster lee sedol. https://www.theguardian.com/technology/2016/mar/15/googles-alphago-seals-4-1-victory-over-grandmaster-lee-sedol, 2016. accessed June 05, 2018.