

UNIVERSITY OF FRIBOURG

MASTER THESIS

Exploration of novel Methods in Reinforcement Learning using Black-Box-Optimization

Author:
Corina Masanti

Supervisor:
Dr. Giuseppe Cuccu

Co-Supervisor:
Prof. Dr. Philippe
Cudré-Mauroux

January 01, 1970

eXascale Infolab
Department of Informatics

Abstract

Corina Masanti

*Exploration of novel Methods in Reinforcement Learning using
Black-Box-Optimization*

Neural networks as generic function approximators can solve many challenging problems. However, they can only be applied successfully for a suited problem structure. In specific, neural networks require differentiability. But there are many areas where calculating an accurate gradient is non-trivial, including problems in Reinforcement Learning (RL). In contrast, Black-Box Optimization (BBO) techniques are less limiting. They presume no constraints on the problem structure, the model, or the solution. With this flexibility, we can study alternative models to neural networks that are unexplored in the context of RL. This thesis aims to achieve pleasing results with a function approximator other than neural networks. I analyze promising models optimized with BBO methods.

Problem -> Solution -> Results

Keywords: Black-Box Optimization, Reinforcement Learning

Contents

Abstract	iii
1 Introduction	1
1.1 Black-Box Optimization	1
1.1.1 Evolution Strategies	1
1.1.2 Covariance Matrix Adaptation Evolution Strategy	1
1.2 Alternative Function Approximators	1
1.2.1 Polynomial	1
1.2.2 Fourier	1
1.2.3 Bézier	1
1.3 Previous Work	1
1.3.1 Benchmarks in Reinforcement Learning	1
2 Experiments	5
2.1 Experiments	5
2.2 Results	5
3 Conclusion	7
3.1 Conclusion	7
3.2 Future Work	7
Bibliography	9

List of Figures

1.1	Reproduced Plots	3
-----	----------------------------	---

Chapter 1

Introduction

Your introduction chapter here.

1.1 Black-Box Optimization

1.1.1 Evolution Strategies

Evolution Strategies (ES) ...

1.1.2 Covariance Matrix Adaptation Evolution Strategy

Covariance Matrix Adaptation Evolution Strategy (CMA-ES) ...

1.2 Alternative Function Approximators

1.2.1 Polynomial

1.2.2 Fourier

1.2.3 Bézier

1.3 Previous Work

1.3.1 Benchmarks in Reinforcement Learning

When developing a novel algorithm, we have to discover how it performs compared to existing methods. For this evaluation, we need standard benchmark problems. These are a set of standard optimization problems. OpenAI Gym ¹ is a toolkit created for exactly this scenario. It contains a collection of benchmark problems with various levels of difficulty. However, not all benchmark problems are meaningful for the evaluation of an algorithm. If a problem is trivial to solve, the results do not reflect the quality of the model adequately.

In the paper *Analyzing Reinforcement Learning Benchmarks with Random Weight Guessing* (Oller, Glasmachers, and Cuccu (2020)), the authors analyze and visualize the complexity of standard RL benchmarks based on score distribution. They tested their approach on the five Classic Control benchmark problems from the OpenAI Gym interface: CartPole, Acrobot, Pendulum, MountainCar, and MountainCarContinuous. Given an RL environment, the authors conducted a fixed series of experiments. For these experiments, they used three NN architectures ($N_{architectures} = 3$): a network without any hidden layers (0 HL), a network with a single hidden layer of 4 units (1

¹gym.openai.com

HL, 4 HU), and a network with two hidden layers of 4 units each (2 HL, 4 HU). They cover a variety of network models that are suitable to solve the task. The evaluation should be as objective as possible and should not include bias in the data. To achieve this, the authors did not include any learning opportunity for the network models. Instead, they chose the network weights i.i.d. from the standard normal distribution $\mathcal{N}(0, 1)$ with Random Weight Guessing (RWG). This approach assures randomness and no directed learning. With this, they initialized 10^4 samples ($N_{\text{samples}} = 10^4$) with different random weights. The amount of samples is intentionally chosen high to determine whether they could see any statistically relevance. Each of these samples of a neural network represents a controller that maps observations to actions in the environment. However, the goal of this thesis is to use an alternative function approximator representing the controller. I will explore this approach later in this thesis. In the paper, the controllers were tested on each environment for 20 independent episodes ($N_{\text{episodes}} = 20$). For each episode, they saved the score in the score tensor S . Algorithm 1 illustrates the procedure with pseudocode.

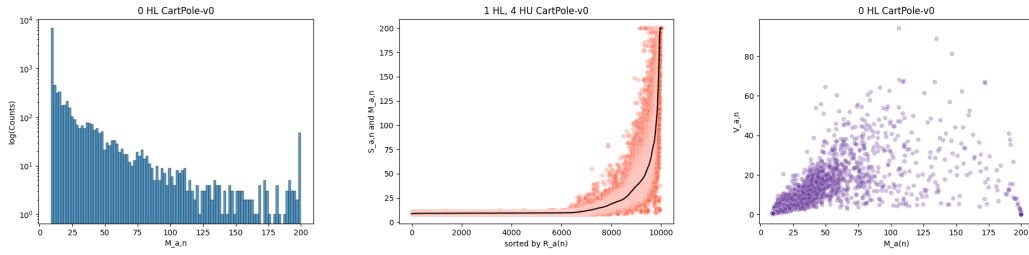
Algorithm 1 Evaluation process taken from Oller, Glasmachers, and Cuccu (2020)

- 1: Initialize environment
 - 2: Create array S of size $N_{\text{architectures}} \times N_{\text{samples}} \times N_{\text{episodes}}$
 - 3: **for** $n = 1, 2, \dots, N_{\text{samples}}$ **do**
 - 4: Sample NN weights randomly from $\mathcal{N}(0, 1)$
 - 5: **for** $e = 1, 2, \dots, N_{\text{episodes}}$ **do**
 - 6: Reset the environment
 - 7: Run episode with NN
 - 8: Store accrued episode reward in $S_{a,n,e}$
-

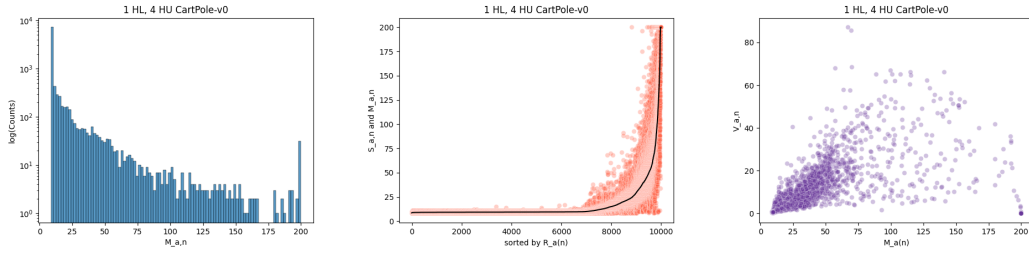
After the authors obtained the scores, they calculated the mean performance over all episodes from a sample and its variance. These entities are important because they give us a measure on how stable the controllers are in an environment. If a controller is able to consistently solve an environment with simple weight guessing, implies that the task is too trivial to take serious as an evaluation measurement. The samples were ranked according to their mean score. They then visualized their results with three plots: a log-scale histogram of the mean scores, a scatter plot of the sample scores over their rank, a scatter plot of score variance over the mean score. I reproduced their results following the mentioned methodology. My findings for the environment CartPole are displayed in Figure 1.1.

Describe method -> results -> conclusion

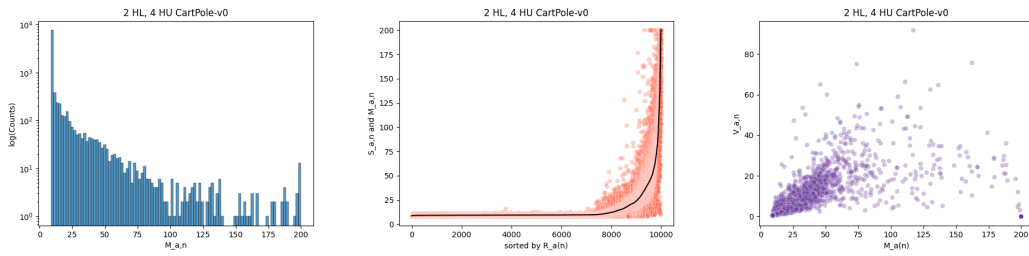
Describe plots, maybe add plots from other environments? now not really meaningful to show all architectures (not enough difference)



(A) Results of network architecture without hidden layers



(B) Results of network architecture with one hidden layer



(C) Results of network architecture with two hidden layers

FIGURE 1.1: **Results.** Results illustrated as histogram and scatter plots.

Chapter 2

Experiments

2.1 Experiments

2.2 Results

Chapter 3

Conclusion

3.1 Conclusion

In this work we...

3.2 Future Work

The continuation of this work includes...

Bibliography

Oller, Declan, Tobias Glasmachers, and Giuseppe Cuccu (Apr. 16, 2020). “Analyzing Reinforcement Learning Benchmarks with Random Weight Guessing”. In: *arXiv:2004.07707 [cs, stat]*. arXiv: 2004.07707. URL: <http://arxiv.org/abs/2004.07707> (visited on 12/07/2021).