UNIVERSITÉ DE FRIBOURG
UNIVERSITÄT FREIBURG

eXascale Infolab

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

Boulevard de Pérolles 90 • 1700 Fribourg • Switzerland
phone +41 (26) 300 84 65 • diuf-secr@unifr.ch • www3.unifr.ch/inf

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
1.2 Impact of bias 4

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, it is important to compare our results with existing models. 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 too trivial to solve, the results do not reflect the quality of the model adequately. We do not need to put a large amount of effort into the creation of a complex model for an easy-to-solve task.

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 neural network architectures

[1] `gym.openai.com`

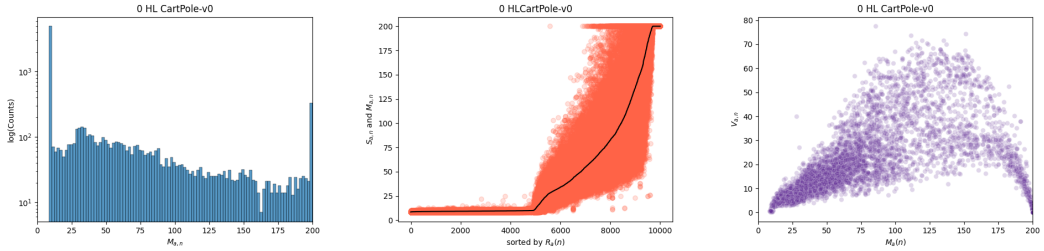
($N_{architectures} = 3$): a network without any hidden layers (0 HL), a network with a single hidden layer of 4 units (1 HL, 4 HU), and a network with two hidden layers of 4 units each (2 HL, 4 HU). With these, they cover a variety of network models that are suited to solve the given tasks. The evaluation should be as objective as possible and should not include bias in the data. To achieve this, the authors did not include any learning opportunities 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. The goal of the paper was not to further analyze the network models but to investigate the benchmark problems themselves. With this in mind, they initialized 10^4 samples ($N_{samples} = 10^4$) with different random weights. The number of samples would be too large for a reasonable learning strategy. However, the large number of samples serves a different purpose than optimizing the results. Instead, the aim is to draw statistical conclusions. Each of these samples of a neural network represents a controller that maps observations to actions in the environment. Later in this thesis, I will explore function approximators other than neural networks representing the controller. In the paper, the authors tested the controllers for each environment during 20 independent episodes ($N_{episodes} = 20$). 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)

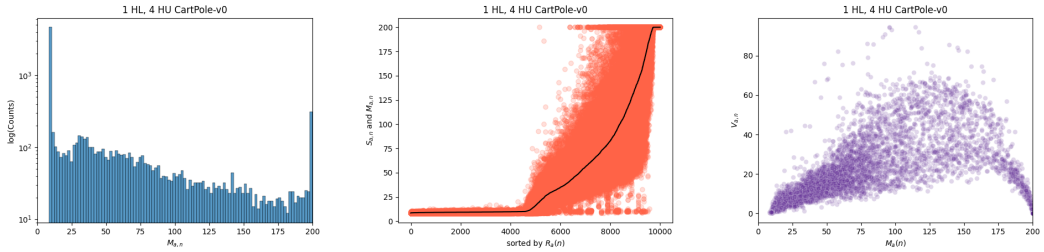
```
Initialize environment
Create array S of size N_architectures × N_samples × N_episodes
for n = 1, 2, ..., N_samples do
    Sample NN weights randomly from N(0, 1)
    for e = 1, 2, ..., N_episodes do
        Reset the environment
        Run episode with NN
        Store accured episode reward in 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 statistics are significant insights. They can reveal how stable the network models are in completing a given task. A low mean value suggests that, in general, the network cannot complete the task. The variance gives us further insight into the score distribution. It illustrates how spread out the scores are from their respective mean score. A high value means that we have high variability. A controller is valuable if it can solve a specific task reliably and stable. Therefore, we strive for a high value for the mean and a low value for the variance. However, training a network with random weight guessing should generally not result in a stable controller. If this is the case, we can assume that the task to solve was too trivial and is not valuable for evaluation measurements. In the illustrations of the paper, the authors ranked the samples according to their mean scores. 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.

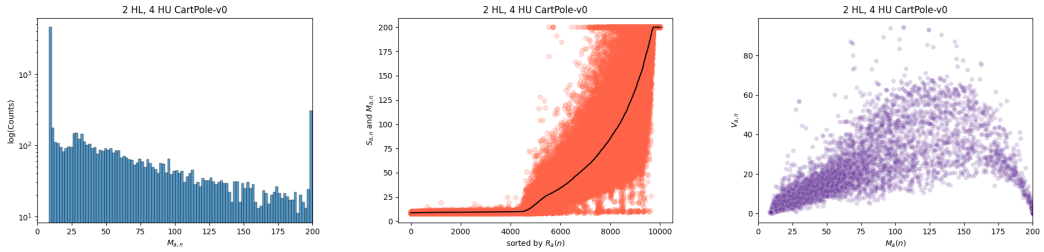
I reproduced the results of the authors following the mentioned methodology. My findings for the environment `CartPole` are displayed in Figure 1.1. The figures illustrate the results for each of the three network architectures. Each row shows the histogram of the mean score values in the left image, the scatter plot of all scores over their rank in the image in the middle, and the scatter plot of the score variance over the mean score in the right image for a specific network architecture. There are



(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 of the benchmark evaluation.** Results of the three network architectures illustrated as: (left images) a log-scale histogram of the mean scores, (middle images) a scatter plot of the sample scores over their rank, (right image) a scatter plot of score variance over the mean score

few differences, but overall all network architectures deliver similar insights. The histogram plots show that the majority of networks receive a low score. Since the weights of the networks were chosen with RWG, this is rather unsurprising. But there is still a significant amount of networks that were able to achieve a high mean value or even the maximum value of 200. With a score of 200, the network was able to solve the task each episode. Therefore, the network could reliably solve the task without any learning technique involved. This should not be the case for a complex task. Furthermore, in the scatter plot in the middle, we can see that the line plot of the mean scores is a continuous increasing line without any jumps. Thus, a suited RL algorithm should generally be able to learn the task incrementally without converging into a local optimum. At the top of the scatter plot, we can see quite a few data points with a score of 200 that have a relatively low mean score. This indicates that a network that generally performs poorly can still solve the task with the right initialization conditions. Lastly, in the scatter plot on the right, we can see the distribution of the variance according to the mean value. On the left side, we have low scores of variance corresponding with a low mean value. These networks were consistently unable to achieve a high score. Without any training involved, we can expect most networks to be in this area. However, in the middle of the plot, the

data points are spread out. For a high variance, the scores of a network differ highly from the mean value. Thus, we might get lucky and receive a high score depending on initialization conditions, but we might as well get a low score. These networks are inconsistent and unstable. On the right side of the scatter plot, we can see that the data points with a high mean value are mostly of low variance. Thus, to achieve a high mean value, the network needs consistency.

Interestingly, the usage of the bias had a relatively large impact on the performance of the network for the environment `CartPole`. Without bias, the networks seem to achieve overall better scores. The plots in Figure 1.1 illustrate the results without bias. For comparison, Figure 1.2 shows the results of a network with two hidden layers with the same configurations as before but this time including bias. In the paper, the authors conducted experiments with and without bias connection for all environments. They discovered that the amount of top performers increases when dropping bias for all five environments.

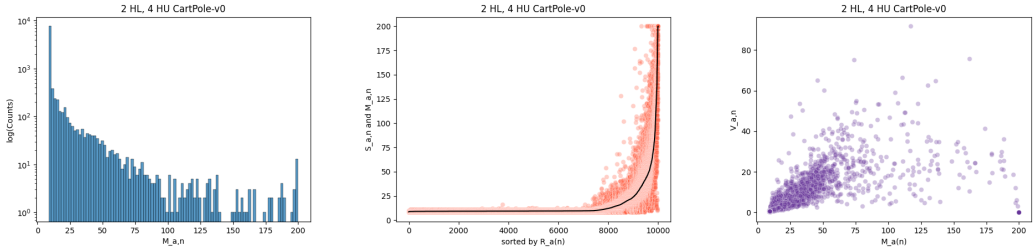


FIGURE 1.2: **Impact of bias.** The figures show the performance of a network with two hidden layers with the same settings as before but here we include bias.

Describe method -> results -> conclusion
Maybe add plots from other environments? adjust images to the ones without using bias, only without bias images in last figure

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).