REPRODUCING: COMBINING MODEL BASED AND MODEL FREE RL VIA MULTI-STEP CONTROL VARIATES

Shashank Kedia, Aahlad Chandrabhatta
University of Michigan, Ann Arbor, MI
{kedia,achandr}@umich.edu

ABSTRACT

We inspect the validity of results and conclusions of the paper titled **Combining Model Based and Model Free RL via Multi-Step Control Variates** submitted to 2018 International Conference on Learning Representations. The paper uses TRPO algorithm by Schulman et al. (2015) as the baseline and compares performance over 4 MuJoCo (Todorov et al. (2012)) environments. We discuss our findings for the baseline results provided and then inspect the main algorithm of the paper.

1 INTRODUCTION

Reinforcement learning algorithms are largely classified into model based and model free algorithms. Model based algorithms learn the transition probabilities between states and use this learned model to find a policy. Model free ones, on the other hand, learn a policy without learning the transition probabilities. The paper makes an attempt at combining model based and model free updates to produce low variance, low bias estimates. In the past there have been attempts to combine the two. The proposed work is influenced by Nagabandi et al. (2017) and is similar to the work of Heess et al. (2015), called the stochastic value gradient algorithm. However, the novelty of the work is in the usage of multi-step prediction through the use of RNN and dynamic rollouts.

2 METHODOLOGY

TRPO algorithm on Swimmer, Reacher, Hopper and Walker environments from MuJoCo are used as baseline. Code corresponding to specific implementations of both the baseline and the proposed Multi-Step Control Variates(MSCV) algorithm is not directed to in the paper. Several hyperparameters are mentioned and the ones not provided have been assumed.

The main aim is to replicate results presented in Table 1 and Figures 1, 4 and 5 of the original paper and to check if TRPO can perform better on other hyperparameters which make it comparable to the MSCV implementation. The original paper compares performance by setting a threshold on the reward for each of these environments and running the algorithms until they reach this pre-set threshold.

We attempt to replicate the baseline results for all the four environments - Swimmer, Reacher, Hopper and Walker. All experiments are run using a publicly available implementation of TRPO from rllab: `https://github.com/Breakend/RLSSContinuousControlTutorial`. Hyperparamter tuning was performed on all four environments to see if the set threshold could be reached in fewer steps than declared by the original paper. We set the maximum number of iterations to 150 for all environments except Walker, for which we set it to 300. We tested on 4 step size values for each environment. Other related information is presented in the appendix.

3 RESULTS

3.1 BASELINES

Hyperparameter tuning was done on all the 4 mentioned MuJoCo environments as OpenAI Brockman et al. (2016) environments and none of our attempts have reached the threshold in the within the number of iterations presented in the paper. On Hopper, none of our runs reached the value of 2000 in under 150 iterations. On Reacher, two of our runs achieved the threshold of -7, but at iteration numbers 64 (Figure 1) and 72, which are beyond said iteration number of 42. On Swimmer, 3 of the runs reached a value of 90, with the earliest at iteration 50(Figure 2), which again is beyond the said iteration number of 30. Finally, for Walker, our best performing TRPO model on walker achieved a value of 1201(Figure 3), leaving the threshold value of 2500 in 279 iterations a far fetched target.

Given more time, more values for hidden states, batch sizes, and maximum iteration numbers would have been explored.

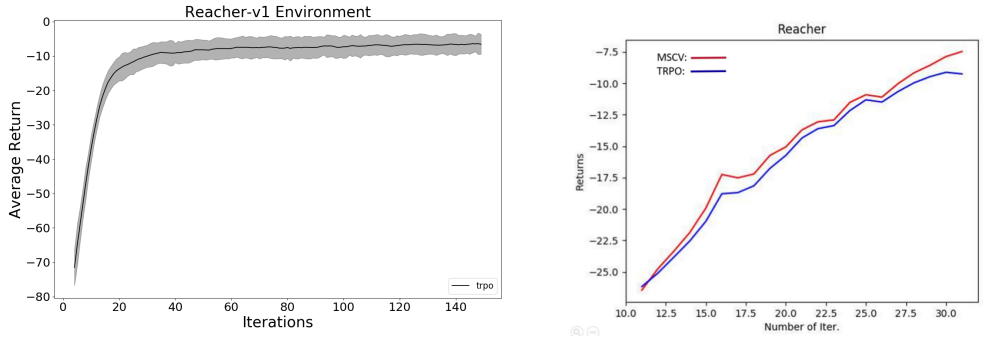

Figure 1: (Left) Our best run on REACHER with TRPO. (Right) The original paper's REACHER graph

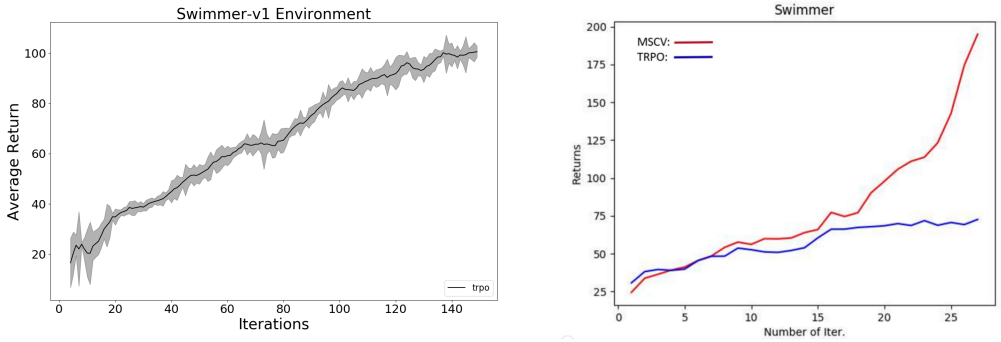

Figure 2: (Left) Our best run on SWIMMER with TRPO. (Right) The original paper's SWIMMER graph

3.2 MSCV

We have not implemented our version of the algorithm. Instead, in this section, we present the challenges we faced while attempting to replicate the work. While the authors do provide limited hyperparameter values and the network structure, considering replication of the algorithm, quite a few key pieces of information are missing.

The networks for policy and value function are provided, but not for the forward model. The paper mentions an RNN is used to train the forward model, but the specifics of RNN, like the activation

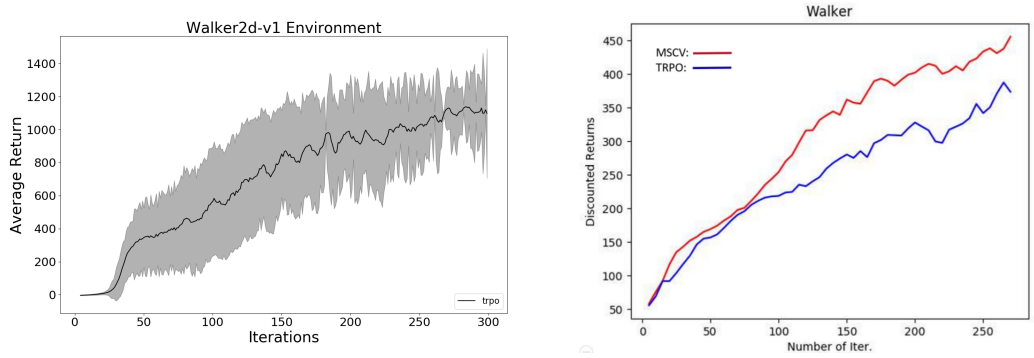

Figure 3: (Left) Our best run on WALKER with TRPO. (Right) The original paper's WALKER graph

functions or the number of steps to unroll to train the model, are not provided. The loss function for the RNN is malformed and requires an assumed rewards model. The predicted rewards from this reward model are compared with imaginary rewards showing a close correlation. Though convergence criterion for training value function V and forward model f are provided to be 'run for n iterations', the value of n is not provided, and neither are the sample sizes for training the models. The policy network update is confusing as it mentions to update the parameters with gradients with TRPO which in its own does not make sense and convergence criterion is also missing.

4 Conclusion

An investigation into the choice of hyperparameters by the authors shows necessary information required for reproducibility to be absent. The graphs provided do not consider long term behaviour of the algorithms. Variance estimates for the rewards received are missing. The baselines results could not be reproduced on parameter tuning. Threshold choices are not explained. Since the proposed algorithm develops on SVG, it would have made a better choice for baseline and would have provided insight into the trade off between the additional complexity of the algorithm and the received rewards.

While the paper brings forth some very interesting ideas, from the view point of reproducibility of results it does fall short on a lot of counts.

References

Greg Brockman, Vicki Cheung, Ludwig Pettersson, Jonas Schneider, John Schulman, Jie Tang, and Wojciech Zaremba. Openai gym. *CoRR*, abs/1606.01540, 2016. URL `http://arxiv.org/abs/1606.01540`.

Nicolas Heess, Greg Wayne, David Silver, Timothy P. Lillicrap, Yuval Tassa, and Tom Erez. Learning continuous control policies by stochastic value gradients. *CoRR*, abs/1510.09142, 2015. URL `http://arxiv.org/abs/1510.09142`.

Anusha Nagabandi, Gregory Kahn, Ronald S. Fearing, and Sergey Levine. Neural network dynamics for model-based deep reinforcement learning with model-free fine-tuning. *CoRR*, abs/1708.02596, 2017. URL `http://arxiv.org/abs/1708.02596`.

John Schulman, Sergey Levine, Philipp Moritz, Michael I. Jordan, and Pieter Abbeel. Trust region policy optimization. *CoRR*, abs/1502.05477, 2015. URL `http://arxiv.org/abs/1502.05477`.

E. Todorov, T. Erez, and Y. Tassa. Mujoco: A physics engine for model-based control. In *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 5026–5033, Oct 2012. doi: 10.1109/IROS.2012.6386109.

5 APPENDIX

5.1 BASELINES RESULTS

Across all environments, for TRPO, we used two hidden layers of size 64 each, with ReLU activation units. For Walker environment, we set the batch size as 25000, while for all other environments, we set it to 5000. These values are assumed from the original paper. The paper originally uses a discount factor of 0.995. Any hyperparameters not mentioned are defaulted to the values provided in rllab implementation.

Table 1: Parameters tuned on for TRPO

HYPERPARAMETERS/GAMMA	**VALUES**
Learning Rate	0.005, 0.01, 0.05, 0.1
Discount factor	0.5, 0.75, 0.9, 0.995

Table 2: Best performing gamma and hyperparameter values and iteration numbers to reach threshold for each environment on TRPO

Environment	Learning Rate	Discount Factor	Threshold	Iterations to reach threshold
Hopper	0.05	0.995	2000	N/A
Reacher	0.05	0.995	-7	64
Swimmer	0.1	0.995	90	52
Walker	0.01	0.995	2500	N/A