Bidirectional Rollouts in Model-Based Reinforcement Learning

Project Background

The use of deep neural networks in reinforcement learning, also known as deep reinforcement learning, has recently achieved remarkable successes. For instance, by combining deep neural networks with Q-learning, a deep reinforcement learning agent is able to achieve human-level and beyond human-level play in Atari games (Mnih et. al., 2013). In robotics, using convolutional neural networks has been rather successful too in learning complex control policies for visuomotors (Levine et. al., 2016). This combination appears to be promising and may be a right path in materializing the eventual goal of having general learning agents living among us.

Despite all the successes, most of the works rely on model-free reinforcement learning. These methods often require huge number of samples in order to solve a task. In other words, they often have very high sample complexity. A model-free agent normally needs millions and millions of interactions, with the environments to achieve high performances. As an often-used and intuitive but likely unfair comparison, we as humans do not require years of non-stop playing to learn how to perform well in a video game which remains to be the case for many of our best model-free reinforcement learning agents. To handle the sample complexity problem, model-based reinforcement learning has been regarded as a potential solution by learning the model of the environment.

With model-based reinforcement learning, the goal is to reduce the number of agent's interactions with the environment. This is done by learning a model that predicts the next state and reward given a state-action pair (Sutton and Barto, 2018). The model is learned by data collected from interactions with the environment. With an accurate model, 'Imagined' data or transitions can be generated from the model without actually interacting with the environment. This can be thought as humans imagining or replaying what could happen in our heads if we take certain actions without actually taking those actions. The use of these imagined transitions to learn is commonly known as planning. In fact, evidence of model-based reinforcement learning has been found in animals (Doll et. al., 2012), pointing to the possibility for artificially intelligent agent to learn efficiently with less samples if it can learn with a model.

To my best knowledge, the first fully formulated model-based reinforcement learning approach is the Dyna architecture proposed by Sutton (1990). In the work, a model generates next states and rewards based on states visited to minimize the number of real-world samples needed. These 'imagined' transitions are then used by the agent to learn from. Such process 'imagines' one step into the future from a state, is formally known as a one-step rollout. By learning from 'imagined' data and real data, it was shown that an agent can learn with way less data. Many works came along after Dyna with a renewed interest over the years as the successes of using neural networks in reinforcement learning also highlighted the issue of sample complexity. For instance, [6] applied model-based approach in deep reinforcement learning on complex locomotion tasks by training a model using randomly generated trajectories. Their promising results indicate the potential of model-based approach in reducing sample complexity.

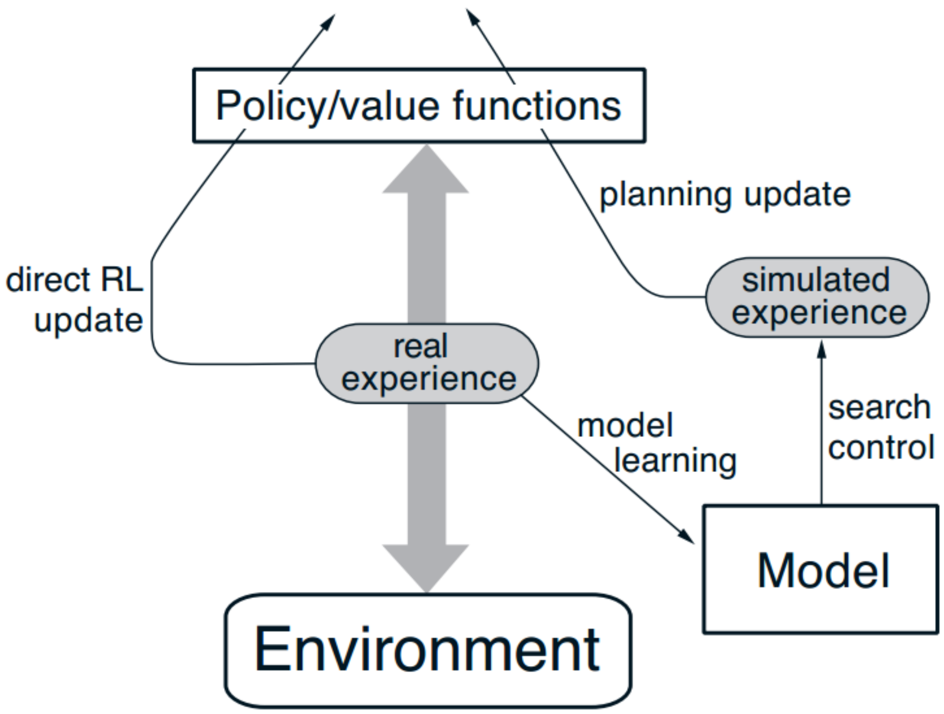

Figure 1: The Dyna Architecture (Sutton, 1990)

Given the potential of model-based reinforcement learning in improving the learning and data efficiency of an artificial agent, there are still many unexplored areas and unresolved issues in the field. The mechanism of rolling out is one of them. Assuming we have a good enough model, how we use the agent to roll out or generate 'imagined' data may matter a lot. Consider humans learning with internal models, a rather loosely connected analogical example, we don't simply imagine one step into the future. Sometimes, we imagine many steps into the future.

Other times, we imagine backwards into the past to hypothesize what might be some other ways that would have led us to this point. Hence, wouldn't it matter for an artificial agent too if it can vary in its mechanism of rollout in terms of the extent and directionality?

As shown by Holland et. al. (2018), the extent of rollout matters a lot. The planning shape, which refers to the extent of a rollout, has significant impact to the effectiveness of model-based learning. An interesting conclusion of the work is the trivial benefits of one-step rollouts and the significantly greater benefits of medium-length rollouts. In addition to the extent, Goyal et. al. (2018) and Ashley et. al. (2018) demonstrated the benefits of rolling out backwards with a backward dynamics model. By rolling out backwards, values can be propagated faster for high reward states, which is especially beneficial for sparse reward environments. With the effects of the extent and directionality of rollouts demonstrated separately, we are motivated to systematically study and understand the separate and combined effects in controlling both the extent and directionality of rollouts. We hypothesize such understanding would allow developments of more dynamic rollout mechanisms, which may greatly amplify the benefits and effectiveness of model-based reinforcement learning.

Project Objectives

Motivated by all the previous works in the rollout mechanisms, the project aims to further the understanding of rollout mechanisms and hopefully provide insights in terms of how our agents should behave during planning with rollouts.

To be more specific, as pointed out above, the extent and directionality may matter a lot in the effectiveness of using a model. Therefore, we would like to explore and answer several questions as our objectives:

- Under what circumstances would forward rollouts be more beneficial than backward rollouts?
- Under what circumstances would backward rollouts be more beneficial than forward rollouts?
- Motivated by Holland et. al. (2018), in which they showed the importance of planning shape in forward rollouts, does the same apply to backward rollouts?
- As a more general and all-encompassing question, how does the extent and directionality of rollout mechanisms interact?
- Can we come up with a rule of thumb in rollout mechanisms that take advantage of both extent and directionality of rollouts?
- Can we automate the decision-making process of rollout mechanisms dynamically that would result in better performance and sample complexity?

By answering these questions, we believe we would be able to provide useful insights and understanding that will be beneficial to the model-based reinforcement learning community, as a contribution to further explore the potential of using a model in reinforcement learning problems.

Project Methodology

As elaborated in the last section, the project aims to improve our understanding in the effect of extent and directionality in rollout mechanisms in model-based reinforcement learning. Based on these insights, we aim to propose more effective rollout mechanisms that can further demonstrate the power of model-based reinforcement learning in reducing sample complexity.

To ground the scope of the research project, we will adapt the Dyna architecture (Sutton, 1990) in which the agent learns from both environment interactions and 'imagined' experiences produced from a learned model. We will vary the rollout mechanisms for our purpose.

Intuition and Ideas on Bidirectional Rollouts

In terms of directionality, we hypothesize forward and backward may be serving different complementary purposes. One way to look at it, is to have the agent to roll out forward to discover novel states and roll out backward whenever interesting states are discovered so the agent can acquire such knowledge quickly.

In terms of extent, we hypothesize, similar to what was found in Holland et. al. (2018), more benefits can be observed also in the backward direction in medium-length rollouts than one-step rollouts.

As for automating these two decisions, there are many possibilities that we would like to explore. One naïve idea would be to treat forward rollout as the exploration process and backward rollout as the value propagation process. In the beginning, when the model is still learning, forward rollouts may be detrimental to the learning of value functions because the imagined next states may simply not exist. On the other hand, the same situation does not apply to backward rollouts because even if the predecessor does not exist, a wrongful update a non-existent predecessor would not affect the value function for the actual states that much. Thus, we may simply use forward rollouts for exploration in the beginning while backward rollouts can be done all the time with more of it when a high reward state is reached. Another possibility would be to frame these decisions as a reinforcement learning process so the agent would be learning how much and which direction to 'imagine' or roll out. Many signals can be used like the model errors, rewards and rollout rewards.

Experimental Plan

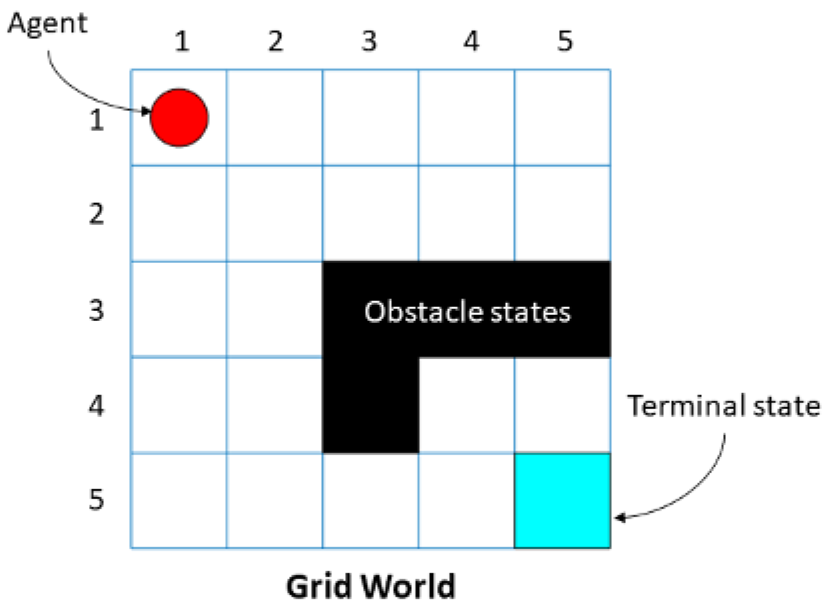

Figure 2: An example of a grid world environment

To address our research questions, we plan to conduct a systematic study on rollout mechanisms. We aim to begin with a small environment where we can perform large scale experiments without the need of huge computational resources. We plan to use varying sizes of grid worlds with discrete action spaces, similar to the ones used by Edwards et. al. (2018). In these grid worlds, the goal of an agent is to navigate to the goal position with a step cost of -0.1 and a reward of 1 when the goal is reached. We will vary the sizes of these grid worlds to compare the effect of different rollout mechanisms. As a start, we will primarily test different forward rollouts to backward rollouts ratios. Hopefully, certain patterns and insights can be identified.

With the understanding on the extent and directionality of rollout mechanisms established, we aim to further verify the results in a more complicated environment, which we will use the MuJoCo environment (Todorov et. al., 2012), an environment with complex control problems in the continuous state and action space.

Note that in all the experiments mentioned above, we will use different qualities of model, namely almost-perfect models (pretrained with lots of data) and imperfect models (learned concurrently with value functions or policies). We will use the rewards and sample complexity as our performance measures. If time allows, we may further explore the situation when the environment has very sparse rewards.

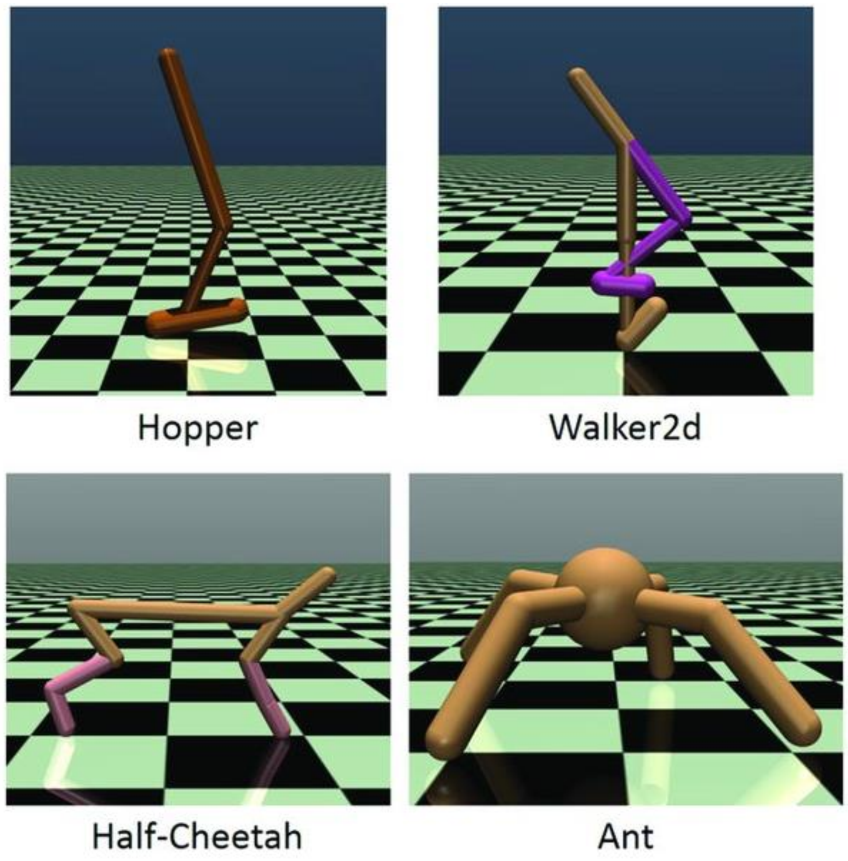

Figure 3: MuJoCo Environments

Programming Tools and Datasets

With the wide array of support in machine learning tools in Python, we will use Python as our primary programming language. As our project will involve the use of neural networks, we will use the PyTorch framework (Paszke et. al., 2017), which will offer us the flexibility and efficiency in performing large scale experiments.

Project Schedule and Milestones

Milestones

30 Sep: Deliverables of Phase 1

- Detailed project plan
- Project Web Page

30 Nov: Experiments on Grid Worlds

- A comprehensive set of experiment results on Grid Worlds, studying the effect of extent and directionality of rollout mechanisms in model-based reinforcement learning
- With the insights and findings established, hopefully, clear patterns should be identified as to how extent and directionality of rollout mechanisms affect performance and sample complexity.
- Ideas on how to automate the decisions on rollout mechanisms should emerge given the initial findings

2 Feb: Deliverables of Phase 2

- Detailed analysis on the initial set of experiments should be conducted
- Additional experiments may be required after the initial set of experiments, they should be done too
- Clean and Comprehensive figures should be prepared
- Detailed Interim Report

17 Mar: Automation of bidirectional rollout mechanisms, experiments on complex environments

- Experiments on ideas to automate bidirectional rollout mechanisms should be done
- Any conclusions should be further confirmed in more complex environments like the MuJoCo environments

19 Apr: Deliverables of Phase 3

- Finalized experimental results and analysis
- Final report

5 May: Project Exhibition

- Exhibition Materials (foam boards, poster, decorations, etc.)

***Official FYP deadlines bolded**

Schedule

17 Sep 2019 - 30 Nov 2019

- Environments and codebase set up for experiments. The goal is to make them modular to facilitate quick experimentation of various ideas and large-scale experiments
- Confirm one single neural network architecture to perform fair and large-scale experiments
- Conduct experiments on Grid Worlds with varying rollout mechanisms to discover insights and the role of extent and directionality in model-based reinforcement learning
- Explore and consolidate on ideas in automating bidirectional rollouts, confirm the ideas mentioned in the intuition and ideas section above

1 Dec 2019 – 2 Feb 2020

- Conduct detailed analysis on initial set of experiments on grid world
- Create comprehensive graphs like learning curves and sample complexity curves
- Review the efficiency in executing the initial set of experiments, improve the codebase
- Prepare codebase to experiment on more complex environments like the MuJoCo environments
- Preparation of the first presentation and interim report
- Invite professors and fellow schoolmates to read my report in order to improve the readability and make sure people without significant background in reinforcement learning can understand

3 Feb 2020 – 17 Mar 2020

- Consolidate and review in a detailed manner of all the potential ideas in automating bidirectional rollouts
- Discuss these ideas with Professor Pan to ensure the direction is reasonable
- Conduct experiments on these ideas
- Any conclusions drawn will be experimented in more complex environments

18 Mar 2020 – 19 Apr 2020

- Conduct more runs to make sure the results are statistically significant
- If time allows, explore the effect of different qualities of model and environments with sparse rewards
- Preparation of final presentation and final report

20 Apr 2020 – 5 May 2020

- Make sure the final report is clear and comprehensive

- Make sure figures are easily understandable
- Preparation of exhibition materials

References

Mnih, Volodymyr, Koray Kavukcuoglu, David Silver, Alex Graves, Ioannis Antonoglou, Daan Wierstra, and Martin Riedmiller. "Playing atari with deep reinforcement learning." arXiv preprint arXiv:1312.5602 (2013).

Levine, Sergey, Chelsea Finn, Trevor Darrell, and Pieter Abbeel. "End-to-end training of deep visuomotor policies." The Journal of Machine Learning Research 17, no. 1 (2016): 1334-1373.

Sutton, Richard S., and Andrew G. Barto. Reinforcement learning: An introduction. MIT press, 2018.

Doll, Bradley B., Dylan A. Simon, and Nathaniel D. Daw. "The ubiquity of model-based reinforcement learning." Current opinion in neurobiology 22, no. 6 (2012): 1075-1081.

Sutton, Richard S. "Dyna, an integrated architecture for learning, planning, and reacting." ACM Sigart Bulletin 2, no. 4 (1991): 160-163.

Nagabandi, Anusha, Gregory Kahn, Ronald S. Fearing, and Sergey Levine. "Neural network dynamics for model-based deep reinforcement learning with model-free fine-tuning." In 2018 IEEE International Conference on Robotics and Automation (ICRA), pp. 7559-7566. IEEE, 2018.

Holland, G. Zacharias, Erin J. Talvitie, and Michael Bowling. "The effect of planning shape on dyna-style planning in high-dimensional state spaces." arXiv preprint arXiv:1806.01825 (2018).

Goyal, Anirudh, Philemon Brakel, William Fedus, Soumye Singhal, Timothy Lillicrap, Sergey Levine, Hugo Larochelle, and Yoshua Bengio. "Recall traces: Backtracking models for efficient reinforcement learning." arXiv preprint arXiv:1804.00379 (2018).

Edwards, Ashley D., Laura Downs, and James C. Davidson. "Forward-backward reinforcement learning." arXiv preprint arXiv:1803.10227 (2018).

Todorov, Emanuel, Tom Erez, and Yuval Tassa. "Mujoco: A physics engine for model-based control." In 2012 IEEE/RSJ International Conference on Intelligent Robots and Systems, pp. 5026-5033. IEEE, 2012.

Paszke, Adam, Sam Gross, Soumith Chintala, Gregory Chanan, Edward Yang, Zachary DeVito, Zeming Lin, Alban Desmaison, Luca Antiga, and Adam Lerer. "Automatic differentiation in pytorch." (2017).