Trade-offs between Model-Free and Model-Based Reinforcement Learning Methods

Ayoub.E Charles. B Nassim. E.M

Laval University
Fall 2022

Abstract

Model-free reinforcement learning (RL) algorithms have gained strong popularity in academia during the past decade due to their ability to learn directly from experience and succeed on a wide variety of tasks. These methods do not require a model of the environment or the underlying dynamics of the system as they can learn directly by trial and errors without performing any planning. This makes them highly flexible but also requiring a lot of data. On the other hand, Model-based RL algorithms can learn a model of the environment and use it to plan and make decisions, which reduces the sample complexity of the learning process as an agent can use multiple world models as simulators for assessing possible outcomes of his decisions. In a way, this can be seen as a "Planning vs Learning" dilemma. Also, the close proximity of model-based methods with cognitive intelligence via the usage of predictive world models ("Common Sense") is making this paradigm attractive for researchers trying to create artificial general agents which can interact and learn a wide set of complex tasks in diverse environments. Although both model-based and model-free RL paradigm offer clear trade-offs in different situation, the boundaries which prescribe which method to use in which situation are still unclear in a certain sense. Therefore, this project was motivated by investigating theoretically those two learning paradigms and assessing the advances made in model based RL like the emergence of methods similar to MuZero [1], which allow the end-to-end learning of simulators via latent models was also something of interest. We finally experimented on a very small scale the performance of those 2 paradigms on a simple RL environment in order to learn how those 2 approaches differs when it comes to their implementation.

Keywords -- *Model-Based RL · Model-Free RL*

I. Introduction

Most of the recent advances in reinforcement learning can be classified as improvements to the Model-Free paradigm, which approximate the optimal policy and/or value function from agent's experiences. But recently, an increase interest in its counterpart, the Model-Based paradigm, started to emerge as the usage of deep neural networks in conjunction with deep lookahead search progressively enlarged the previously narrow domain of successes using those methods. Indeed, by improving how the learning/usage of dynamic model(s) were made for usage in planning, initial working applications expanded from traditional games with defined rules like Go and Chess to a wider set of real-world applications, like discovering algorithm for better matrix multiplications with AlphaTensor [1], chemical retrosynthesis via 3N-MCTS [2], quantum exploration algorithms for the hidden-variable approximation of quantum parameter landscapes [3], and video compression optimization [4]. In fact, many of those innovations make use of deep neural network for guiding an MCTS planning procedure.

But shortcomings of model based RL (MBRL) are still numerous and are well detailed in [5] [6] and to a certain extend in section 3 of this report. But to cite a few, there is the extra computation and memory necessary, the instability due to stochasticity and partial observability planning, the non-stationarity & multi-step prediction, and finally the integration of state and temporal abstraction in the agent's planning. Also, one of the most common problem is called the model bias, which emerges when the policy optimization tends to exploit regions of the state-space where insufficient data is available to train the model, leading to catastrophic failures. In other words, it happens that regions of a simulator are inaccurate, and when optimizing for higher reward policies, an overfitting to the inaccurate simulator is something not desirable, as the learned dynamics model has some kind of bias that an agent will be overfitting to. Recently, methods like Model-Ensemble Trust-Region Policy Optimization (ME-TRPO) were designed to circumvent some of this issue [7].

1.1) Attractivity of Model-Based RL

Nonetheless, many advantages and theoretical principles guide researchers to improve and solve the riddles of model based RL (MBRL). This is in part due to its claimed sample efficiency as it can uses planning to reduce the real-world interaction that is considered a limited resource. The intuition is simple: unlike model-free RL where the agent collects the data to learn directly a value function and/ or a policy, the model-based paradigm proposes a situation where the agent use the data to learn a model of the world first, which allow him to simulate what the world is like, and then can then be used to find a good policy or value function. In this situation, the agent can learn massively without the need to collect new data from the real environment, hence the claimed sample efficiency. Another attractivity of MBRL is the increase optimality of the joint usage of planning and learning, as the local approximation errors that emerge from generalization of a value network can be smoothed locally via planning by looking ahead to states with less uncertain value predictions. For example, the learned value prediction for the current state in the game of Chess might be inaccurate due to the high level of uncertainty, but if we explicitly lookahead, we can find states with a more certain value prediction (win or loss), hence the utility of local planning to assist the learning component in smoothing out the errors in the approximation, which provide better optimality and performance. This relates well to the statements made by Rich Sutton in his essay "The Bitter Lesson", where he explains that the power of scaling computation by search and learning together is a key concept in the design of learning systems.

An additional aspect that makes MBRL attractive is the possibility of transfer learning of either the information content of the learned world models or the world model themselves allow for MBRL agents to speed-up learning to a new context. This can be either in a situation where we have similar dynamics function but different reward function (Ex: New video game level) or when there is a slight change in the transition dynamics (Ex: Sim-to-Real). Finally, the MBRL paradigm is also attractive as it may represent a solution for the safe learning and deployment of RL Agents on real-world systems via models that can determine safe regions of explorations and make sure that some constraints are respected. For example, in the case of a Self-Driving RL implementation, an agent will always avoid to drive-off a cliff as it can make use of a physics model in order to plan the next states and forecast the massive negative reward coming from this action.

1.2) Intelligence via predictive world models

One of the most important features of MBRL is its natural and logical link to human intelligence, which serves as a foundational inspiration for designing artificial general intelligent (AGI) agents able to interact and learn a wide set of complex tasks in diverse environments. Indeed, the usage of predictive world/dynamics models that learn to represent the world and predict it may be the solution to the sample-inefficient model-free RL paradigm, which requires a large amount of data to learn new skills and stand in contrast with human or animal learning. This may be explained by the fact that a simple scalar reward may represent a low-information feedback signal to the learning agent, which makes him require a large sum of trials for optimal learning. A strong analogy to this idea is what we call common sense, which can be seen as a set of world models that guide an agent on what is likely, plausible, or impossible. Those models serve multiple purpose as they can be a foundation for learning new skills in a few-shot manner, predicting the consequences of actions via planning, or produce interpretations of perception that are consistent with the common sense acquired. In fact, an active topic of research nowadays is how should those world models be trained. Is it via the usage of agent's experiences, hence actions and corresponding rewards, or by self-supervised learning using large amount of unlabeled data where the agent train himself to predict the world states? And which objective function should be used? In this situation, many links could be made to the free energy principle by Karl Friston, but as this subject is fascinating but not the main focused of this report, [6] are excellent sources for more information on the subject. But this said, the case for model-free RL is also very strong. One key innovation that could help tip the balance for model-free RL are the emergence of accurate, complex, and distributed simulations engines like Nvidia Isaac-Sim (Omniverse), which could allow RL agents to train for as many iterations as needed in a short amount of time, and in a parallel way in order to produce Sim-to-Real transfers of the accumulated learning to the real-world. Of course, the modeled distribution must be the same as in the real world, hence the emergence of techniques like domain randomization that can produce strong policies robust to shift in the distribution.

Hence if interaction samples with those simulated environments can become exponentially accessible and attractive due to continuous improvement in the quality of simulator's accuracy and distribution matching, then the problem of sample complexity and access to large amount of data may be considered partly resolved. But as long as uncertainty will subsist in the decision-making, planning will be a key component of future intelligent systems. Therefore, the boundary between situations where planning is necessary and how much of it is needed is still an active topic of research and relates closely to the question of whether to use Model-Free or Model-Based RL when faced with a specific problem. Indeed, although planning may be useful for policy improvement, generating experiences to learn from, and acting directly when evaluation comes, some proposes that it is most useful in the learning process for computing learning targets and generating data. Other interrogations like the design of the search procedure are also topic to deeper analysis. On that account, this project was motivated by investigating those two learning paradigms which are model-based and model-free RL. Moreover, assessing the advances made in model based RL like the emergence of methods similar to MuZero that allow the end-to-end learning of Planning and Transition dynamics using latent models was also something of interest. Consequently, the question of whether those kinds of approach can compete with modern Model-Free RL methods could be quantified via experiments in this project.

II. Problem Description

Following the theoretical context presented in the introduction, the decided objective was to explore the performance trade-offs between different methods coming from the Model-Based and the Model-Free RL Paradigm on a simple environment. A discrete game with a low degree of complexity was chosen due to our limitations in time and resources, as an environment with high complexity will requires large amount of computing resources to train a modern model-based architecture with large action and state spaces. Furthermore, keeping a fast experimentation feedback loop was important for us in order to assess the attractivity of different methods.

- Connect4

The selected game was Connect4, which is a popular 2 player game using a vertical board with 6 rows and 7 columns where the objective is to be the first player to form a horizontal, vertical, or diagonal line of four using some or all of the 21 discrete tokens given to each player (See Figure __). Consequently, there is a maximum of 42 total playable positions. Also, the first player is chosen at random at each initialization of a game. Importantly, this game is considered solved as it can be proven that if the first player plays the center column and plays perfectly, they cannot lose. On the other hand, if the first player does not choose the center column and the second player plays perfectly, the ladder can always draw. [__]

- MDP Formalism: We consider the game as a Markov Decision Process (MDP) M defined by the tuple {S, A, P, R, γ} where S is the set of states, A is the set of available state-dependent actions, R: S × A → R is a bounded reward function, P : S × A × S → [0, 1] is the transition function where P (s' |s, a) is the probability of transitioning to state s' from s when action a was taken. And finally, γ ∈ [0, 1] is the discount factor. We consider a policy π : S × A → [0, 1] as a mapping from state to a distribution over actions where π (a|s) denotes the probability of taking action a in state s.

The optimal policy is the function that gives the best action a in all states s ∈ S. Thus, the goal of the different RL algorithms investigated will be to obtain a policy which maximizes the expected sum of discounted rewards provided by our MDP. In other words, an optimal Connect4 strategy is the sequence of actions that maximizes the probability of winning the game for a specific player. The state space can be seen as the state of the board at time t with all the played pieces from the two players. Of course, the assumption of gravity limits the number of possible state configurations an agent can see. When it comes to the action space, an action is simply the act of selecting one of the 7 columns where a token will be placed.

III. Selected Approach

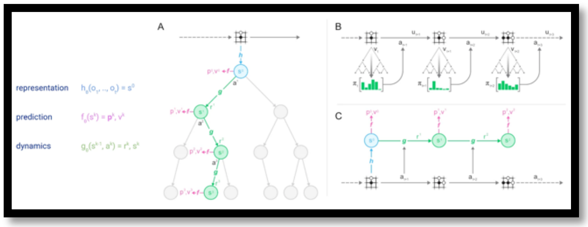
- Model-Free Approaches: Model-Free approaches are often segmented according to multiple categories, whether they are On-policy or Off-policy, or whether they are Value-based Methods or Policy-based Methods (or both), or whether their update is of the Monte-Carlo type or more of the Temporal Difference type. But in general, they can all be seen as trial-and-error learners which rely on learning a mapping [S_t --> A_t] or [S_t --> Values (S_t, A_t)] that maximize the total reward R_t obtained on the given environment. The 3 model-free methods selected for training on the Connect4 environment were DQN (a Value Method), PPO (a Policy Gradient Method) and A2C (an Actor-Critic Method).

- Model-Based Approaches: Methods in this category rely primary on planning for taking decisions. This is done by learning via experience or accessing directly given transition models, which traditionally are represented as mapping from **(St, At) --> Rt+1** and **(St, At) --> St+1.** For the state transition model, three main types of functions might be considered: Forward model with a mapping (st, at) → st+1, reverse model with st+1 → (st, at), or an inverse model with (st, st+1) → at. Most of those mappings are learned via function approximators like Neural Networks, often trained through supervised learning. Therefore, if we assume the state space and action space known, a model M is a representation of an MDP {S, A, P, R} parametrized by η where M = Pη, Rη, representing a state transition and reward approximators Pη ≈ P & Rη ≈ R. Two main classes of MBRL methods are often considered: the **decision-time planning** approaches and the **background planning** approaches. While the former uses the model to select actions, the ladder employs the model for updating the policy. Moreover, we can say that Integrated Architectures (or End-to-end Model-based RL), can represent a third class which combines both ideas where a parameterized differentiable model and a parameterized differentiable planning procedure are jointly optimized at the same time.

- MuZero:

Model & Learning: Unlike its predecessor, the planning in MuZero is done in a hidden state space that uses a learned model μ_θ parameterized by θ. It is composed of three main functions: representation, predictions, and dynamics. The representation function h (Blue on the figure) maps from a set of observations (Connect4 board) to the hidden state s used by the neural network. The dynamics function g (Green on the figure) represents the mapping from a state s to the next state s_{t+1} based on an action a_{t+1}. It is also responsible for predicting the reward from this transition. And the prediction function f estimates values for the policy and value in function of the current state. Also, the policy network p (s, a) corresponds to a probability distribution over all actions from a state while the value network v(s) estimates the probability of winning from the current state. Now as the model receives an observation, it transforms it into a hidden state that is updated iteratively via a recurrent process that receives the previous hidden state and a hypothetical next action. At each step of this process, the role of the model is to predict the policy (action to choose), value function (predicted winner), and the immediate reward (value of a specific action). For that, the model needs to be trained end-to-end to predict those 3 quantities via minimizing the errors with the policy/value estimates (generated by search) and the observed reward.

Search & Action-Selection: Like AlphaGo and AlphaZero, MuZero uses Monte Carlo Tree Search (MCTS) to aggregate neural network predictions and choose which actions to apply. MCTS is an iterative, best-first tree search procedure with three main phases: simulation, expansion and backpropagation. Those 3 steps are executed many times in order to produce a search tree over future action sequences one node at a time. Now after the search phase is completed, an action is selected according to the statistics of the search and the action is applied to the environment so the game can proceed to the next state with the given rewards. Action selection can either be greedy or with a degree of exploration according to whether we select the action with the most visits or whether we sample an action proportional to its visit count n (s, a) after applying a temperature parameter for controlling the exploration-exploitation. Finally, data from the search and environment is added to a replay buffer {o, a, r^{env}, π^{MCTS}, v^{MCTS}}.



- Figure __: (A) Schematic representation of a search tree made by MCTS with the corresponding predictions made by MuZero. (B) Episode generation. (C) Episode training.

IV. Experimentation Settings

No space left for adding content in this section... (Maximum 4 pages).

V. Results & Discussion & Conclusion

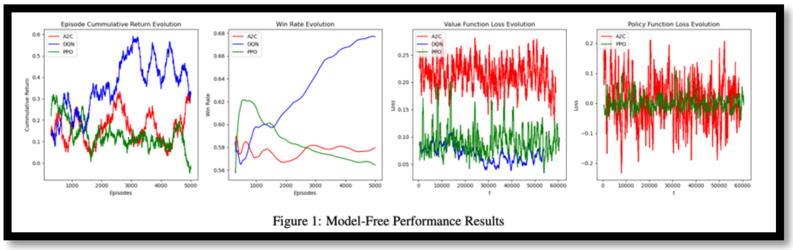


Figure 1: Model-Free Performance Results

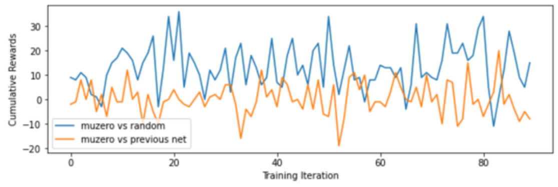


Figure 2: Model-Based Performance Results

Model-free methods, DQN, A2C and PPO, were trained by playing against a random agent for 5 000 episodes and compared based on the evolution of their respective loss function, cumulative returns, and win rate. For all methods, the value function and policy function, if applicable, were modeled as Convolutional Neural Networks (CNN), given the space dependency inherent to the board game representation as a grid. DQN was trained with a learning rate of 10^{-3}. For A2C, critic and actor learning rates respectively 10^{-4}, and for PPO were respectively 10^{-8} and 10^{-6}. The PPO approach was trained using 8 actors and updated over 8 epochs for each optimization step.

To conclude, we quickly realized that our results were much more limited and could not be used to assess the performance trade-offs between model-based and model-free methods. Indeed, problems which requires strong planning are more complex that connect4 and those are example of environment we should choose in order to make a better comparison of those two paradigms. Also, implementing a version of MuZero called *Reanalyze* should be something to explore as it was designed to be more data efficient than the current MuZero implemented here.

VI. References

[1] Fawzi, A., Balog, M., Huang, A. et al. Discovering faster matrix multiplication algorithms with reinforcement learning. Nature 610, 47–53 (2022).

[2] Segler, M., Preuss, M. Waller, M. Planning chemical syntheses with deep neural networks and symbolic AI. Nature 555, 604–610 (2018).

[3] Dalgaard, Mogens et al. "Global optimization of quantum dynamics with AlphaZero deep exploration." npj Quantum Information 6 (2019).

[4] Mandhane, Amol, et al. "Muzero with self-competition for rate control in vp9 video compression." arXiv preprint arXiv:2202.06626 (2022)

[5] Sutton, Richard S and Barto, Andrew G "Reinforcement learning: An introduction. MIT Press (2018)

[6] A Path Towards Autonomous Machine Intelligence, Yann LeCun. Courant Institute of Mathematical Sciences, New York University. June 27, 2022