Does using advantage function rather than state-action value function in the planning part of model-based reinforcement learning reduce the variance?

Amir Noohian [*1] **Alireza Isavand** [*2] **Reza Abdollahzadeh** [*2]

Abstract

Model-based reinforcement learning (MBRL) methods, particularly those grounded in model predictive control (MPC), employ environment models for pre-planning actions before executing them. MPC optimizes a scoring function to determine optimal actions. While incorporating global information in the scoring function accelerates the learning process, it introduces variance in learning. This project report aims to address this issue by proposing the use of the sum of advantage functions (Sum-Advantage) as a scoring function, in contrast to the previously employed sum of state-action values (Sum-Value). Our experiments on one Gym environment show that our method overcomes all the previous methods in terms of the return variance.

1. Introduction

Reinforcement learning (RL) methods can be classified as model-free RL (MFRL) and model-based RL (MBRL). In contrast to MFRL, MBRL methods are usually more sample-efficient, as they iteratively approximate the model of the environment and use the model to plan actions (Sutton & Barto, 2018). One of the common planning methods in continuous environments is model predictive control (MPC), a technique that optimizes a scoring function over a finite time horizon. Typically, the optimization in MPC is solved using the cross-entropy method (CEM), a gradient-free optimization method. CEM samples numerous trajectories composed of state-action sequences with an horizon (H) and evaluates each trajectory with a scoring function. The output of the optimization process is a locally optimal policy or sequence of actions (up to the horizon H) (Hewing et al., 2020).

The scoring function for evaluating trajectories determines the information used to plan actions. Scoring functions in the literature can be classified into three types. The first one, named Sum-Reward, sums the immediate reward of each state-action along the trajectory $\sum_{t=0}^{H} \gamma^t r(s_t, a_t)$ (Sikchi et al., 2022; Nagabandi et al., 2017), where H is the planning horizon, and $\gamma \in [0, 1)$ is the discount factor. To add global information in evaluating trajectories, an alternative scoring function, named Sum-Reward-Value, sums the immediate reward of each state-action along the trajectory and adds the estimated value of the terminal state $\sum_{t=0}^{H-1} \gamma^t r(s_t, a_t) + \gamma^H \hat{V}(s_H)$ (Lowrey et al., 2018; Morgan et al., 2021; Sikchi et al., 2022; Hoeller et al., 2020; Zong et al., 2021; Hansen et al., 2022; Charlesworth & Montana, 2021). Finally, the last scoring function, named Sum-Value, sums the estimated discounted value of each state-action along the trajectory $\sum_{t=0}^{H} \gamma^t \hat{Q}(s_t, a_t)$ (Raisi et al., 2022). Sum-Value was proposed to include more information about the future in evaluating trajectories, making it efficient in environments with sparse rewards and densely distributed states.

Although Sum-Value outperforms the other scoring functions in terms of sample efficiency and average return, it imposes variance in learning according to Raisi et al. (2022). In this study, we aim to modify the Sum-Value scoring function to reduce its variance. To do so, we want to use the estimated advantage function rather than the estimated state-action value function in the scoring function. We define the scoring function as $\sum_{t=0}^{H} \gamma^t \hat{A}(s_t, a_t)$ and call it Sum-Advantage. The research question we want to answer in our study is as follows:

Does using the advantage function rather than the state-action value function in the planning part of MBRL reduce the variance?

As we know, the advantage function is defined as $A(s_t, a_t) = Q(s_t, a_t) - V(s_t)$, where $Q(s_t, a_t)$ is the action-state value function, and $V(s_t)$ is the state value function. Using the advantage function, we add a baseline, which is the state value function, to the previous scoring function,

[*]Equal contribution [1]Department of Electrical and Computer Engineering, University of Alberta, Edmonton, Canada [2]Department of Computing Science, University of Alberta, Edmonton, Canada. Correspondence to: Amir Noohian <noohian@ualberta.edu>.

Proceedings of the 39^{th} International Conference on Machine Learning, Baltimore, Maryland, USA, PMLR 162, 2022.

the action-state value function. As it is usually claimed that using a baseline reduces the variance (Greensmith et al., 2004), we expect that the variance of the learning part is reduced. Moreover, by using the advantage function rather than reward, we benefit from the global information in the scoring function. Therefore, the previous scoring function does not lose its property of using the information from the future.

The proposed scoring function is integrated into an MBRL algorithm, which is composed of two parts: MPC for planning and MFRL for learning. Furthermore, in this study, we only focus on the planning part. Thus, we assume that the model of the environment is known, and there is no model learning. To the best of our knowledge, it is the first time that the advantage function is used in the planning part of MBRL.

The remainder of this report is structured as follows: Section 2 describes the background needed for this research project. Section 3 contains details of the scoring function and the methodology that we incorporated. In section 4, we will review the experimental settings used for this project and the results our experiments brought out. In section 5 we conclude the main key points of the paper and suggest future works. Finally, in section 6, we remark on the main contributions of each team member.

2. Background

2.1. Reinforcement Learning

In the context of reinforcement learning, a Markov decision process (MDP) can be defined as $< S, A, R, P, \gamma >$, where:

- S represents the state space,
- A represents the action space,
- $R : S \times A \to \mathbb{R}$ represents the reward function,
- $P : S \times A \times S \to [0, 1]$ represents the transition probability, and
- $\gamma \in [0, 1]$ is the discount factor.

The policy $\pi : S \times A \to [0, 1]$ defines the probability of taking an action in a given state. In this environment, the return $G_t = \sum_{t=0}^{\infty} \gamma^t r_t$ represents the sum of discounted rewards.

The state value, denoted as $V_\pi(s)$, is defined as the expected return starting from state s and following the policy π:

$$V_\pi(s) = \mathbb{E}_\pi \left[\sum_{t=0}^{\infty} \gamma^t r(s_t, \pi(s_t)) \mid s_0 = s \right]. \quad (1)$$

Additionally, the state-action value, denoted as $Q_\pi(s, a)$, is defined as the expected return starting from state s, taking action a, and then following the policy π:

$$Q_\pi(s, a) = \mathbb{E}_\pi \left[\sum_{t=0}^{\infty} \gamma^t r(s_t, \pi(s_t)) \mid s_0 = s, a_0 = a \right]. \quad (2)$$

2.2. Planning with Model Predictive Control

In model predictive control (MPC), instead of solving an infinite-horizon optimization problem, we solve a sequence of finite-horizon optimal control problems with a horizon of H to find the optimal action sequence:

$$\pi_{MPC}(s_t) = \arg \max_{\pi_{t:t+H-1}} \mathbb{E}_\pi \left[\sum_{i=0}^{H-1} \gamma^i r(s_{t+i}, a_{t+i}) + \gamma^H r_f(s_{t+H}) \right], \quad (3)$$

where $a_t = \pi_t(s_t)$, and the function r_f quantifies the reward for the truncated time, for instance, the state value function of the last state.

To efficiently solve the path-control problem, an effective sampling method is essential. Thijssen & Kappen (2015), suggests utilizing the optimal policy. A possible approach is employing the cross-entropy method, an adaptive importance sampling technique. This method involves iteratively enhancing a distribution, denoted as p, to estimate the optimal sampler. The objective is to minimize the cross-entropy between the optimal sampler p^* and the current sampler p.

In the cross-entropy method (CEM), N samples $\{(a_{j,0}, a_{j,1}, ..., a_{j,H-1})\}_{j=1}^{N}$ are generated using the current sampler p. Each $a_{j,i}$ represents the action taken at each time step. Subsequently, N trajectories are created by applying this action sequence to the transition model.

Next, the value of each trajectory is evaluated, and the sampler distribution is adjusted to fit k samples with the maximum value using Maximum Likelihood Estimation (MLE), as expressed by:

$$\phi' = \arg \max_{\phi} \sum_{j=1}^{N} \mathbb{1}(v(\tau_j) \geq v_{th}) \log p_\phi(\tau_j), \quad (4)$$

where $\mathbb{1}$ represents the indicator function, and v_{th} denotes the value of the trajectory with the k-th highest value.

After finding the optimal sequence, the agent takes the first action, and then the process is repeated again for the next state.

2.3. Soft Actor-Critic (SAC)

The soft actor-critic (SAC) algorithm (Haarnoja et al., 2018) is a model-free, online, off-policy, actor-critic reinforcement

learning method. SAC extends the traditional actor-critic architecture by introducing a soft approach to optimization, incorporating entropy regularization into the learning process.

The key components of SAC include the actor network, which maps states to actions, and the critic network, which evaluates the goodness of those actions. The policy learned by the actor is augmented with an entropy term, encouraging stochasticity in action selection. This stochasticity helps prevent premature convergence to suboptimal policies, promoting effective exploration in the learning process.

The objective function in SAC combines the standard expected return with an entropy term. The entropy term encourages the policy to explore more diverse actions, striking a balance between exploration and exploitation.

Mathematically, the objective function for SAC is defined as follows:

$$J(\pi) = \mathbb{E}_{(s_t,a_t)\sim D}\left[\sum_{t=0}^{\infty}\gamma^t\left(r(s_t,a_t) + \alpha\mathcal{H}(\pi(\cdot|s_t))\right)\right], \tag{5}$$

where π is the policy, D is the replay buffer, r is the reward function, α is the temperature parameter, and $\mathcal{H}$ is the entropy of the policy.

3. Method

In this section, we propose the scoring function and its integration into the planning part of MBRL.

3.1. Scoring Function

As mentioned in Section 1, the use of the estimated state value function to evaluate trajectories in the planning part of MBRL introduces variance in the learning phase. To address this issue, we propose using the estimated advantage function. Thus, we propose the discounted summation of the estimated advantage function to score trajectories as:

$$S = \sum_{t=0}^{H}\gamma^t\hat{A}(s_t,a_t), \tag{6}$$

where the estimated advantage values over a trajectory with the planning horizon H are summed up to determine the trajectory score S. We refer to the proposed scoring function as Sum-Advantage.

3.2. Advantage Summation MBRL

This subsection outlines the incorporation of the Sum-Advantage scoring function into MBRL. The algorithm's framework aligns with the MBRL algorithm proposed by Raisi et al. (2022), but differs by utilizing Sum-Advantage instead of Sum-Value. This algorithm, named Advantage Summation MBRL, consists of two parts: planning and learning. Planning is done with MPC. After receiving the current state s_t from the environment, the model $\hat{f}(s_t,a_t)$ is used to sample several trajectories with the planning horizon H from the policy π_θ. The collected trajectories are then scored by the Sum-Advantage scoring function, and the first action from the highest-scored trajectory is executed in the environment. To learn the state and state-action values and policy, we use an actor-critic off-policy algorithm for the learning part. We adopt the soft actor-critic (SAC) (Haarnoja et al., 2018) to benefit from the exploration induced by the soft policy updates based on the maximum entropy principle.

A description of Advantage Summation MBRL can be found in Algorithm 1. Using the environment model $\hat{f}(s_t,a_t)$ with the planning horizon H, the agent shoots N trajectories at each time step (line code: $6-17$). Eq. (6) is applied to assess each pair $(s_{t:t+H}, a_{t:t+H})$ (lines $12-13$), allowing the evaluation of the corresponding trajectory. Data from the environment model is accumulated in the replay buffer D_1, subsequently utilized to train the critic and value networks (line 16). The first action from the best-scored trajectory is then executed in the environment (line code: 19). The real data is collected in the replay buffer D_2, and it is then employed to train both the policy network and the critic and value networks (line 20).

4. Experiments

4.1. Experimental Setups

In the following section, we will describe the experimental framework that was employed and compare two scoring functions as our reference benchmarks. This comparison will illustrate the effect of the proposed method on the learning variance:

1. **Sum-Reward**: First introduced by Chua et al.(2018), it is computed by summing up the discounted rewards, given as $\sum_{t=0}^{H}\gamma^t r(s_t,a_t)$.
2. **Sum-Reward-Value**: Similar to the previous method with the addition of discounted the estimated value of the terminal state, which is given as $\sum_{t=0}^{H-1}\gamma^t r(s_t,a_t) + \gamma^H Q^\pi(s_H,a_H)$, applied in Feinberg et al.(2018).
3. **Sum-Value**: First introduced by Raisi et al. (2022), it is computed by summing up the estimated state-action values along each trajectory with respect to the horizon H, given as $\sum_{t=0}^{H}\gamma^t\hat{Q}(s_t,a_t)$.
4. **Sum-Advantage**: Our new scoring function, com-

puted by adding a baseline of estimated state values to the previous method, resulting in a summation of estimated state-action advantage functions, given as $\sum_{t=0}^{H} \gamma^t \hat{A}(s_t, a_t) = \sum_{t=0}^{H} \gamma^t (\hat{Q}(s_t, a_t) - \hat{V}(s_t))$.

Algorithm 1 Advantage Summation MBRL

Given $f(s_t, a_t)$: real environment, $\hat{f}(s_t, a_t)$: model, D_1: model replay buffer, D_2: environment replay buffer, $\pi_\theta(s_t)$: actor-network, $Q_\phi(s_t, a_t)$: critic network, $V_\psi(s_t)$: value network γ: discounted factor;
H: Horizon length;
N: Number of trajectories;
while Task Not Completed **do**
  $s_t \leftarrow CurrentState$
  for $n \leftarrow 0$ to N **do**
    $s_0 = s_t$
    $S(n) = 0$
    for $h \leftarrow 0$ to H **do**
      $a_h = \pi_\theta(s_h)$
      $s_{h+1} = \hat{f}(s_h, a_h)$
      $A_h = Q_\phi(s_h, a_h) - V_\psi(s_h)$
      $S(n) += \gamma^h A_h$
      $Store\ (s_h, a_h)\ in\ D_1$
    end for
    $Use\ D_1\ to\ Update(Q_\phi, V_\psi)$
  end for
  $a_t = \underset{a_{t:t+H}}{\operatorname{argmax}}\ S$
  $s_{t+1} = f(s_t, a_t)$
  $Store\ (s_t, a_t)\ in\ D_2$
  $Use\ D_2\ to\ Update(\pi_\theta, Q_\phi, V_\psi)$
end while

Our primary focus is to develop a better scoring function for the planning step in model-based reinforcement learning (MBRL). To narrow down the key factors affecting this goal, we assume that we have previously constructed a unique model of the environment. Rather than learning a new model based on the environment, we leverage the real environment dynamics.

The architecture of SAC consists of one actor and two critic networks, each constructed with three layers of fully connected networks using ReLU (Agarap, 2019) activation functions for hidden layers. The minimum of q-values output from the two critics is incorporated to calculate the critic value, similar to the approach taken in van Hasselt et al.(2015). Additionally, we train a value network to approximate state values, which, along with the critic value, is used to calculate the scoring function. The value network similarly consists of three layers of fully connected networks with ReLU as the activation function for hidden layers.

We determine the mean performance for each scoring method by averaging the returns of each evaluation step across corresponding experiments. Subsequently, we calculate the standard deviation based on the variance of the performance over the last 30,000 steps for each scoring method.

Our experiments are conducted in the Gym environment Hopper-v3 (Brockman et al., 2016), a two-dimensional environment where a single-legged agent aims to master the task of hopping forward based on feedback from position, velocity, and angular velocity observations. This environment, chosen for its relative simplicity, allows for a greater number of experiments and has exhibited strong performance with the previous state-of-the-art method or the Sum-value approach. This makes it a suitable choice for testing our new method. Our goal is to preserve the best aspects of the previous performance while simultaneously reducing variance on a higher and more challenging scale within this environment.

We used the average episode reward as the primary performance metric across these runs. The evaluation was conducted in a single environment with 5 runs for each scoring method. The agent's performance was assessed every 1000 steps, accumulating a total of 150,000 steps to capture a comprehensive measure of its experience. In our experiments, the batch size was set to 256, and the network layers consisted of 256 nodes each. During the rollout, we generated 4 paths each time with a horizon of 4 transitions.

Method	Best Performance	Std	Mean Performance
Sum-Advantage	3234.59	135.48	3118.27
Sum-Value	3311.79	164.20	3199.45
Sum-Reward	3371.04	538.31	2508.94
Sum-Reward-Value	3428.32	344.51	3193.57

Table 1. Return and standard deviation comparison for four different scoring methods.

4.2. Results

The table 1 shows the standard deviation and performance for each scoring function. Although the performance of our method is comparable with other methods, the standard deviation of our method stands superior to all other methods.

Figure 1 displays the overall model performance for Sum-Value, Sum-Advantage, Sum-Reward, and Sum-Reward-Value. The shaded areas represent the range formed by adding and subtracting one standard deviation from the mean of the model performance across runs for each scoring function, with the mean depicted as the line in between. According to this figure, while our model shows comparable performance with Sum-Value, both are inferior in comparison with the Sum-Reward-Value method. Meanwhile, the variance of our method is notably lower than others in the figure throughout the learning process.

Despite the uncertainty stemming from single-environment experiments, the results in this environment show promise, highlighting the potential for further exploration of our method in different environments to form a more decisive opinion on the behavior of Sum-Advantage.

5. Conclusion

In this project, we initiated a novel exploration within the domain of model-based reinforcement learning (MBRL) by introducing the concept of using the sum of advantage functions (Sum-Advantage) as a scoring function in the planning phase. The results on the Hopper environment show that using Sum-Advantage as the scoring function reduces the standard deviation of the average return compared to Sum-Value, Sum-Reward, and Sum-Reward-Value as scoring methods while maintaining a comparable average performance. It is noteworthy that our findings can benefit from further experiments across diverse setups to thoroughly evaluate the effectiveness of all aspects of the Sum-Advantage approach. The implementation of the proposed method is available on GitHub at github.com/re0078/advantage_sum_mbrl. We encourage interested readers to explore the codebase for a more in-depth understanding of the implementation details.

Looking forward, the integration of the advantage function into the planning phase of MBRL presents a promising avenue for future research and refinement. This exploratory study illuminates the potential of Sum-Advantage without conclusive assertions, emphasizing the necessity of additional experiments to validate and extend our initial observations.

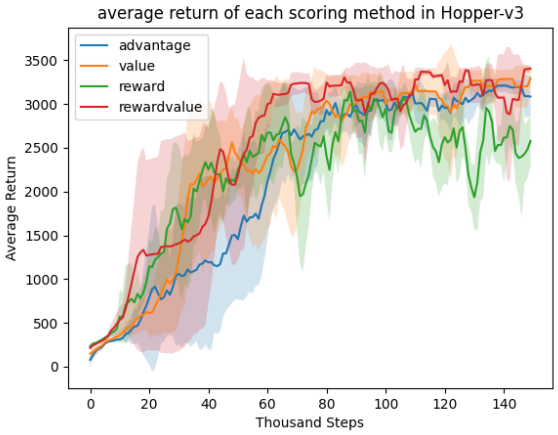

Figure 1. Performance comparison of the advantage summation method in model-based reinforcement learning algorithm with other scoring functions.

6. Contribution

The contribution of each member is stated as follows: Amir Noohian mainly worked on reviewing the literature, finding the gap in the related works, and explaining the methodology. Alireza Isavand's contribution to the project includes an in-depth study and comprehensive explanation of the necessary preliminary concepts and theories. Reza Abdollahzadeh worked on designing and situational adjustment of experimental settings. Infrastructure preparation, running experiments, and visualization, reporting, and analysis of results are parts of the contributions of all team members.

References

Agarap, A. F. Deep learning using rectified linear units (relu), 2019.

Brockman, G., Cheung, V., Pettersson, L., Schneider, J., Schulman, J., Tang, J., and Zaremba, W. Openai gym. *arXiv preprint arXiv:1606.01540*, 2016.

Charlesworth, H. J. and Montana, G. Solving challenging dexterous manipulation tasks with trajectory optimisation and reinforcement learning. In *International Conference on Machine Learning*, pp. 1496–1506. PMLR, 2021.

Chua, K., Calandra, R., McAllister, R., and Levine, S. Deep reinforcement learning in a handful of trials using probabilistic dynamics models, 2018.

Feinberg, V., Wan, A., Stoica, I., Jordan, M. I., Gonzalez, J. E., and Levine, S. Model-based value estimation for efficient model-free reinforcement learning, 2018.

Greensmith, E., Bartlett, P. L., and Baxter., J. Variance reduction techniques for gradient estimates in reinforcement learning. *Journal of Machine Learning Research*, 5 (9), 2004.

Haarnoja, T., Zhou, A., Abbeel, P., , and Levine, S. Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor. In *International Conference on Machine Learning*, pp. 1861–1870. PMLR, 2018.

Hansen, N., Wang, X., and Su, H. Temporal difference learning for model predictive control. *arXiv preprint arXiv:2203.04955*, 2022.

Hewing, L., Wabersich, K. P., Menner, M., and Zeilinger, M. N. Learning-based model predictive control: Toward safe learning in control. *Annual Review of Control, Robotics, and Autonomous Systems*, 3:269–296, 2020.

Hoeller, D., Farshidian, F., and Hutter, M. Deep value model predictive control. In *Conference on Robot Learning*, pp. 990–1004. PMLR, 2020.

Lowrey, K., Rajeswaran, A., Kakade, S., Todorov, E., and Mordatch, I. Plan online, learn offline: Efficient learning and exploration via model-based control. *arXiv preprint arXiv:1811.01848*, 2018.

Morgan, A. S., Nandha, D., Chalvatzaki, G., D'Eramo, C., Dollar, A. M., and Peters, J. Model predictive actor-critic: Accelerating robot skill acquisition with deep reinforcement learning. In *International Conference on Robotics and Automation*, pp. 6672–6678. IEEE, 2021.

Nagabandi, Kahn, G., Fearing, R. S., and Levine, S. Neural network dynamics for model-based deep reinforcement learning with model-free fine-tuning. *arXiv preprint arXiv:1708.02596*, 2017.

Raisi, M., Noohian, A., Mccutcheon, L., and Fallah, S. Value summation: A novel scoring function for MPC-based model-based reinforcement learning. *arXiv preprint arXiv:2209.08169*, 2022.

Sikchi, H., Zhou, W., and Held, D. Learning off-policy with online planning. In *Conference on Robot Learning*, pp. 1622–1633. PMLR, 2022.

Sutton, R. S. and Barto, A. G. *Reinforcement learning: An introduction*. MIT press, 2018.

Thijssen, S. and Kappen, H. Path integral control and state-dependent feedback. *Physical Review E*, 91(3):032104, 2015.

van Hasselt, H., Guez, A., and Silver, D. Deep reinforcement learning with double q-learning, 2015.

Zong, T., Sun, L., and Liu, Y. Reinforced iLQR: A sample-efficient robot locomotion learning. In *International Conference on Robotics and Automation*, pp. 5906–5913. IEEE, 2021.