Policy Gradient with Second Order Momentum

Tianyu Sun [1]

Abstract

In this project, we propose an enhancement to traditional policy gradient methods in reinforcement learning by incorporating second order momentum. Our approach aims to improve convergence and exploration-exploitation trade-offs by introducing momentum alongside methods like REINFORCE. We derive the formulation of second order momentum based on existing literature and integrate it into REINFORCE, observing its impact on convergence. We also explored higher order policy gradient algorithms and bias reduction techniques on the augmented model. Overall, this project contributes to advancing the understanding and application of policy gradient algorithms in reinforcement learning, particularly in optimizing convergence and exploration-exploitation dynamics through the integration of second order momentum.

1. Introduction

Reinforcement Learning (RL) has emerged as a powerful paradigm for solving sequential decision-making problems in various domains, ranging from robotics to game playing. Among RL algorithms, policy gradient methods have garnered significant attention due to their ability to directly optimize policy parameters to maximize cumulative rewards. Traditional policy gradient methods, such as REINFORCE, have shown success in a wide range of applications. However, they often suffer from slow convergence and inefficiencies in exploration-exploitation trade-offs.

To address these limitations, recent research has explored the integration of momentum techniques, inspired by developments in optimization algorithms, into policy gradient methods. Momentum has been proven effective in accelerating convergence and improving exploration-exploitation dynamics in optimization problems. In this context, we propose an extension to traditional policy gradient methods by incorporating second order momentum. Building upon existing work, we aim to enhance the convergence speed and exploration capabilities of RL agents by leveraging second order information.

We begin by introducing the theoretical foundation of second order momentum in policy gradient methods, drawing insights from recent literature. Subsequently, we outline our methodology, which includes implementing and benchmarking traditional policy gradient algorithms on standard RL environments, followed by the integration of second order momentum corrections. We also discuss the computational challenges and various variance reduction technique to improve the stability of the policy.

Overall, this research contributes to the ongoing efforts in advancing RL algorithms by leveraging momentum techniques to optimize policy learning in complex environments.

2. Background and related work

Several studies have explored the integration of momentum-based techniques into policy gradient methods to address the shortcomings of traditional approaches. For instance, (Shen et al., 2019) introduced the concept of Hessian-aided policy gradient, leveraging second-order information to enhance optimization performance. Their work demonstrated improved convergence properties compared to standard policy gradient methods.

Building upon this line of research, (Salehkaleybar et al., 2023) proposed a Momentum-Based Policy Gradient with Second-Order Information. Their approach extends momentum-based techniques and used a variance reduction technique called STORM by adaptively updating learning rate based on previous iteration.

Moreover, (Tran & Cutkosky, 2022) explored the utilization of second-order momentum in stochastic gradient descent, demonstrating its efficacy in enhancing optimization performance. Although their work focuses on SGD, the principles of second-order momentum optimization can be adapted and integrated into policy gradient methods to further improve convergence dynamics.

These studies collectively highlight the growing interest in leveraging momentum-based techniques, particularly

[1]Department of Mathematics, Indiana University, Bloomington, Indiana. Correspondence to: Tianyu Sun (ts19@iu.edu).

second-order momentum, to enhance the convergence properties and exploration-exploitation trade-offs of policy gradient methods in reinforcement learning. By integrating these advancements, researchers aim to push the boundaries of RL optimization and pave the way for more efficient and effective learning algorithms.

3. Policy Gradient with Second Order Momentum

3.1. Second Order Policy Gradient Theorem

Let the reward function be defined as

$$J(\theta) = \sum_{s\in\mathcal{S}} d^{\pi}(s)V^{\pi}(s) = \sum_{s\in\mathcal{S}} d^{\pi}(s) \sum_{a\in\mathcal{A}} \pi_{\theta}(a|s)Q^{\pi}(s,a) \tag{1}$$

The the policy gradient theorem states that

$$\nabla J(\theta) \propto \mathbb{E}_{s\sim d^{\pi}, a\sim\pi(s)} \left[Q^{\pi}(s,a)\nabla_{\theta} \ln\, \pi_{\theta}(a|s)\right] \tag{2}$$

And the second order momentum of the policy can be calculated as

$$\nabla^2 J(\theta) \propto \mathbb{E}_{\tau\sim p(\tau;\pi_\theta)}[\nabla_\theta \ln(p(\tau;\pi_\theta))\nabla_\theta \Psi(\theta) + \nabla^2_\theta \Psi(\theta)] \tag{3}$$

where $\Phi(\theta;\tau) = \sum_{h=0}^{H} \ln \pi_\theta(a_h|s_h) \sum_{t=h}^{H} \gamma^t r(s_t,a_t)$ and $p(\tau;\pi_\theta)$ is the distribution of trajectory under the policy π_θ.

Notice that with $p(\tau;\pi_\theta)$ defined as

$p(\tau;\pi_\theta) = \rho(s_0)\Pi_{h=1}^{H} \mathcal{P}(s_{h+1}|s_h,a_h)\pi_\theta(a_h|s_h)$,

then

$$\nabla_\theta \ln(p(\tau;\pi_\theta)) = \nabla_\theta \left[\sum_{h=1}^{H} \ln(\pi_\theta(a_h|s_h))\right] \tag{4}$$

is independent of the transition probability of the trajectory.

3.2. Runge Kutta method

The Runge-Kutta method, a second-order numerical technique commonly used for solving ordinary differential equations (ODEs), involves multiple stages of function evaluation to improve accuracy compared to simpler methods. By computing weighted averages of function evaluations at different points within the step interval, Runge-Kutta captures the dynamics of the system more accurately, making it a valuable tool for approximating solutions to differential equations.

In the context of policy gradient methods, Runge-Kutta provides a more accurate means to estimate the dynamics of the environment or gradients of the value function. This accurate estimation is crucial for calculating gradients of the expected return with respect to policy parameters, enabling RL algorithms to perform informed policy updates for faster convergence and improved performance in complex environments. The basic steps to incorporate Runge-Kutta in policy gradient is shown below

$$\begin{aligned} \tilde{\theta}_t &= \theta^t + \nabla J(\theta^t) \\ \nabla \tilde{J}(\theta^t) &= \mathbb{E}_{s\sim d^{\pi}_{\tilde{\theta}}, a\sim\pi_{\tilde{\theta}}(s)} \left[Q^{\pi}(s,a)\nabla_\theta \ln\, \pi_\theta(a|s)\right] \\ \theta^{t+1} &= \theta^t + \alpha\nabla J(\theta^t) + (1-\alpha)\nabla\tilde{J}(\tilde{\theta}_t) \end{aligned} \tag{5}$$

3.3. Advantage function

The advantage function in reinforcement learning serves as a critical tool for assessing the quality of actions taken by an agent in a given state. It quantifies the advantage or disadvantage of choosing a particular action over others relative to the current policy. Formally, the advantage function $A(s,a) = Q(s,a) - V(s)$ measures the difference between the expected return obtained by taking action a in state s and the expected return obtained by following the current policy.

3.4. Clipping

Clipping the policy gradient is a crucial technique in reinforcement learning to ensure stable and efficient training. By limiting the magnitude of the gradients, typically through techniques like gradient clipping, the algorithm prevents large updates that could lead to unstable learning dynamics or divergence.

3.5. Entropy Regularization

Entropy regularization (Ahmed et al., 2019) is a widely used technique in reinforcement learning that encourages exploration by penalizing overly deterministic policies. By adding an entropy term to the objective function, the algorithm is incentivized to maintain a certain level of uncertainty in its actions, promoting exploration of the environment and preventing premature convergence to suboptimal policies. For policy gradient, the entropy term is defined as

$$\mathbb{H}(\pi(\cdot|s_t)) = \mathbb{E}_{a\sim\pi(\cdot|s_t)[-\ln \pi(a|s_t)]} \tag{6}$$

4. Experiments

4.1. Dataset and implementation details

We only used Cartpole with logistic policy. The implementation on environment with multiple action maybe tricky because of the dimensionality of the hessian softmax. And it is usually easier to find the second derivative using neural network and autodiff.

We trained the policy over 500 episodes and we used total (base REINFORCE), hessian (second order REINFORCE), and rk (Runge Kutta REINFORCE). For each method, we run it 5 times and find the distribution of evaluation given the policy. We tested the effect of baseline, clipping, and entropy on each method. We used a learning rate of 0.002. For clipping, we double the learning rate with a clipping size of 50.

4.2. Results

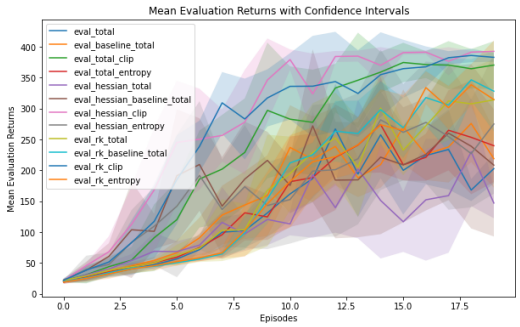

Figure 1. The Runge Kutta method shows the most promising result. And adding the baseline definitely help with improving the performance of the policy

4.3. Ablation Study

Across all models, Runge Kutta provides the best performance. The performance of Hessian policy gradient is improved with the aid of various technique. Adding clipping could drastically improve the performance of policy gradient across all three methods because the learning rate is doubled. But the variance of clipping is also drastically reduced. The effect of entropy and baseline is different across all three methods. Adding entropy or baseline didn't reduce the variance as I expected. But they did improve the performance of the model.

5. Future Works

In future research, extending the proposed approach to handle environments with continuous action spaces holds significant promise for advancing policy gradient methods in reinforcement learning. Integrating second-order momentum with neural network-based policies tailored for continuous action spaces presents a compelling avenue for exploration. Additionally, exploring dynamic adaptation of momentum techniques, such as incorporating ideas from adaptive optimization algorithms like Adam, could further enhance the effectiveness of the approach. By dynamically adjusting momentum parameters based on the training dynamics, the algorithm can adapt more efficiently to changing environments and improve convergence properties.

Table 1. Comparison of Policy Gradient Strategies at the last episode

Model	**Mean Evaluation**	**Std Evaluation**
Policy Gradient	202.70	23.99
Policy Gradient+clip	370.51	38.72
Policy Gradient+entropy	239.90	58.98
Policy Gradient+baseline	218.84	47.17
Hessian Policy Gradient	147.00	24.53
Hessian Policy Gradient+clip	392.79	0.55
Hessian Policy Gradient+entropy	274.99	112.73
Hessian Policy Gradient+baseline	208.43	115.63
Runge Kutta	314.18	66.33
Runge Kutta+clip	383.09	7.15
Runge Kutta+entropy	314.75	95.46
Runge Kutta+baseline	328.21	65.32

References

Ahmed, Z., Le Roux, N., Norouzi, M., and Schuurmans, D. Understanding the impact of entropy on policy optimization. In Proceedings of the 36th International Conference on Machine Learning (ICML), volume 97, pp. 151–160. PMLR, 2019.

Salehkaleybar, S., Khorasani, S., Kiyavash, N., He, N., and Thiran, P. Momentum-based policy gradient with second-order information, 2023.

Shen, Z., Ribeiro, A., Hassani, H., Qian, H., and Mi, C. Hessian aided policy gradient. In International conference on machine learning, pp. 5729–5738. PMLR, 2019.

Tran, H. and Cutkosky, A. Better sgd using second-order momentum. Advances in Neural Information Processing Systems, 35:3530–3541, 2022.