Cartpole problem

DEBAJYOTI MAITY

December 2023

Abstract

In this study, we systematically evaluate variations of model-free policy gradient methods, specifically focusing on the 'vanilla' policy gradient approach. Our investigation involves implementing the gradient of the logarithm of the policy probability with baseline adjustment to mitigate variance. The study aims to provide a comprehensive understanding of the performance of this refined method through extensive experimentation on the cart pole regulator benchmark. The presented approach not only simplifies the original formulation but also contributes to reducing variance in the learning process. The provided Portable C++ code for both plant and algorithms facilitates easy replication of results and encourages future research by allowing the seamless insertion of new algorithms for further exploration in the domain of parameterized policy search algorithms

I.Introduction

The reinforcement learning community has been actively working on establishing standards for evaluating and comparing various reinforcement learning methods. This paper contributes to this effort by focusing on model-free policy gradient methods, a significant area within reinforcement learning. Our objective is to create reliable benchmarks by providing well-defined simulation setups and finely tuned algorithms.

We extensively compare different versions of key policy gradient algorithms, considering various aspects to establish a foundation for evaluating future parameterized policy search algorithms. The emphasis is on offering a solid starting point for researchers developing their algorithms, variants, or exploring different parameter choices. The meticulously designed algorithms and carefully chosen meta parameters (e.g., learning rates, exploration initialization) contribute to the robustness of our baselines.

We address the second goal of the paper by highlighting essential considerations when applying policy gradient methods. To achieve this, we select a well-known and easily accessible control problem: cart-pole regulation in both deterministic and stochastic versions. This choice ensures that the challenges posed by the plant are neither too difficult nor too easy, allowing for a focus on fundamental properties applicable across different domains. The provided

online resources, including software and documentation, facilitate easy access and further exploration of the evaluated algorithms and plants.

II. Policy Gradient Methods: Algorithms & Variants

Policy gradient methods, similar to most other reinforcement learning methods, consist of two steps, i.e., (i) a policy evaluation step which results in an estimate of the gradient $\nabla_\theta J(\theta)$ of the expected return

$$J(\theta) = \mathbb{E}\left[\sum_{t=0}^{N} r_t \mid \pi_\theta\right]$$

for the current policy π_θ (where r_t denotes the reward at time t) and (ii) a policy improvement step which is realized by updating the policy parameters through steepest gradient ascent

$$\theta_{k+1} = \theta_k + \alpha_k \nabla_\theta J(\theta)$$

where α_k denotes a learning rate.

For policy gradient estimation, we consider three different approaches, i.e., finite difference methods (FD) which perturb the policy parameters of a deterministic policy, vanilla policy gradient methods (VPG), and natural policy gradient methods (NG) which both perturb the motor commands by employing a stochastic policy . The three different methods are described in the following sections. However, we will not give an in-depth account of each method but rather present the basics behind each of them and refer to the literature for detailed derivations.

For computing the policy update, we compare standard gradient descent updates with constant learning rates $\alpha_k = \alpha$ with update rules based on the Rprop method. The Rprop method is a variable step-size method based on the sign of the partial derivatives rather than on the magnitude. Thus, it is usually significantly more robust and does not require significant manual tuning.

Reinforcement Learning and Policy Gradient Methods

Alternatively to perturbing the parameters θ_k of a deterministic policy $u = \pi(x)$, one could choose a stochastic policy $u \sim \pi(u \mid x)$ such that $u = \mu(x) + \epsilon$ where ϵ is a perturbation of the nominal motor command $\mu(x)$. In this case, one can make use of the likelihood ratio trick, i.e., if we rewrite Eq.(1) in terms of trajectories τ with probability

$$p(\tau \mid \theta) = p(x_0) \prod_{t=0}^{N-1} p(x_{t+1} \mid x_t, u_t)\pi(u_t \mid x_t)$$

The reward function is defined as: and summed reward $R(\tau) = \sum_{t=0}^{N} r_t$ and differentiate, we obtain

$$\begin{aligned}
\nabla_\theta J(\theta) &= \nabla_\theta \int Rp(\tau \mid \theta)\, d\tau \\
&= \int \nabla_\theta p(\tau \mid \theta) R(\tau)\, d\tau \\
&= \int p(\tau \mid \theta) \nabla_\theta \log p(\tau \mid \theta) R(\tau)\, d\tau \\
&= \mathbb{E}\left\{\nabla_\theta \log p(\tau \mid \theta) R(\tau)\right\}
\end{aligned} \tag{1}$$

as $\nabla_\theta p(\tau \mid \theta) = p(\tau \mid \theta) \nabla_\theta \log p(\tau \mid \theta)$ by definition. If we additionally make use of Eq.(4), we see that

$$\nabla_\theta \log p(\tau \mid \theta) = \sum_{t=0}^{N-1} \nabla_\theta \log \pi(u_t \mid x_t)$$

and thus we can compute the gradient from samples without a model of the system.

$$\mathbb{E}_{s \sim \mu^\pi}\left[R(s)\right]$$

where μ^π is the stationary distribution of the Markov chain for π (on-policy state distribution under π). For simplicity, the parameter θ would be omitted for the policy π when the policy is present in the subscript of other functions; for example, μ and R should be μ^π and R^π if written in full.

Imagine that you can travel along the Markov chain's states forever, and eventually, as time progresses, the probability of ending up in one state becomes unchanged—this is the stationary probability for π. μ^π is the probability that s occurs when starting from s_0 and following policy π for t steps. The existence of the stationary distribution of the Markov chain is one of the main reasons why the PageRank algorithm works.

It is natural to expect that policy-based methods are more useful in continuous spaces. In continuous spaces, there is an infinite number of actions and/or states to estimate values for, making value-based approaches computationally expensive. For example, in generalized policy iteration, the policy improvement step $\theta' \leftarrow \arg\max_{\theta'} Q^\pi(s, a)$ requires a full scan of the action space, suffering from the curse of dimensionality.

Using gradient ascent, we can move θ toward the direction suggested by the gradient ∇_θ to find the best θ for π that produces the highest return.

Computing the Raw Gradient

Using the two tools above, we can now get back to our original goal, which was to compute the gradient of the expected sum of (discounted) rewards. Formally, let $R(\tau)$ be the reward function we want to optimize (i.e. maximize). Using the above two tricks, we obtain:

$$\nabla_\theta E_{\tau\sim\pi_\theta}[R(\tau)] = E_{\tau\sim\pi_\theta}[R(\tau)\cdot\nabla_\theta\left(\sum_{t=0}^{T-1}\log\pi_\theta(a_t|s_t)\right)]$$

In the above, the expectation is with respect to the policy function, so think of it as $\tau\sim\pi_\theta$. In practice, we need trajectories to get an empirical expectation, which estimates this actual expectation.

So that's the gradient! Unfortunately, we're not quite done yet. The naive way is to run the agent on a batch of episodes, get a set of trajectories (call it τ) and update with $\theta\leftarrow\theta+\alpha\nabla_\theta E_{\tau\in\tau[R(\tau)]}$ using the empirical expectation, but this will be too slow and unreliable due to high variance on the gradient estimates. After one batch, we may exhibit a wide range of results: much better performance, equal performance, or worse performance. The high variance of these gradient estimates is precisely why there has been so much effort devoted to variance reduction techniques. (I should also add from personal research experience that variance reduction is certainly not limited to reinforcement learning; it also appears in many statistical projects which concern a bias-variance tradeoff.)

Introduce a Baseline

The standard way to reduce the variance of the above gradient estimates is to insert a baseline function $b(s_t)$ inside the expectation.

For concreteness, assume $R(\tau)=\sum_{t=0}^{T-1}r_t$, so we have no discounted rewards. We can express the policy gradient in three equivalent, but perhaps non-intuitive ways:

$$\text{(i)}\quad \nabla_\theta E_{\tau\sim\pi_\theta}[R(\tau)] = E_{\tau\sim\pi_\theta}\left[\left(\sum_{t=0}^{T-1}r_t\right)\cdot\nabla_\theta\left(\sum_{t=0}^{T-1}\log\pi_\theta(a_t|s_t)\right)\right]$$

$$\text{(ii)}\quad \nabla_\theta E_{\tau\sim\pi_\theta}[R(\tau)] = E_{\tau\sim\pi_\theta}\left[\sum_{t'=0}^{T-1}r_{t'}\sum_{t=0}^{t'}\nabla_\theta\log\pi_\theta(a_t|s_t)\right]$$

$$\text{(iii)}\quad \nabla_\theta E_{\tau\sim\pi_\theta}[R(\tau)] = E_{\tau\sim\pi_\theta}\left[\sum_{t=0}^{T-1}\nabla_\theta\log\pi_\theta(a_t|s_t)\left(\sum_{t'=t}^{T-1}r_{t'}-b(s_t)\right)\right].$$

Comments:

Step (i) follows from plugging in our chosen $R(\tau)$ into the policy gradient we previously derived.

Step (ii) follows from first noting that $\nabla_\theta E_\tau[r_{t'}]=E_\tau[r_{t'}\cdot\sum_{t'=t}^{0}\nabla_\theta\log\pi_\theta(a_t|s_t)]$. The reason why this is true can be somewhat tricky to identify. I find it easy to think of just re-defining $R(\tau)$ as $r_{t'}$ for some fixed time-step t'. Then, we do the exact same computation above to get the final result, as shown in the equation of the "Computing the Raw Gradient" section. The main difference

now is that since we're considering the reward at time t', our trajectory under expectation stops at that time. More concretely, $\nabla_\theta E(s_0, a_0, \ldots, s_T)[r_{t'}] = \nabla_\theta E(s_0, a_0, \ldots, s_{t'})[r_{t'}]$. This is like "throwing away variables" when taking expectations due to "pushing values" through sums and summing over densities (which cancel out); I have another example later in this post which makes this explicit.

Next, we sum over both sides, for $t' = 0, 1, \ldots, T-1$. Assuming we can exchange the sum with the gradient, we get

$$\begin{aligned}
\nabla_\theta E_{\tau\sim\pi_\theta}[R(\tau)] &= \nabla_\theta E_{\tau\sim\pi_\theta}\left[\sum_{t'=0}^{T-1} r_{t'}\right] \\
&= \sum_{t'=0}^{T-1} \nabla_\theta E_{\tau(t')}[r_{t'}] \\
&= \sum_{t'}^{T-1} E_{\tau(t')}[r_{t'} \cdot \sum_{t=0}^{t'} \nabla_\theta \log \pi_\theta(a_t|s_t)] \\
&= E_{\tau\sim\pi_\theta}\left[\sum_{t'}^{T-1} r_{t'} \cdot \sum_{t=0}^{t'} \nabla_\theta \log \pi_\theta(a_t|s_t)\right].
\end{aligned}$$

where $\tau(t')$ indicates the trajectory up to time t'. (Full disclaimer: I'm not sure if this formalism with τ is needed, and I think most people would do this computation without worrying about the precise expectation details.)

Step (iii) follows from a nifty algebra trick. To simplify the subsequent notation, let $f_t := \nabla_\theta \log \pi_\theta(a_t|s_t)$. In addition, ignore the expectation; we'll only re-arrange the inside here. With this substitution and setup, the sum inside the expectation from **Step (ii)** turns out to be

$$r_0 f_0 r_1 f_0 r_2 f_0 \ldots r_{T-1} f_0 + r_1 f_1 + r_2 f_1 + r_2 f_2 + \ldots + r_{T-1} f_1 + r_{T-1} f_2 \ldots + r_{T-1} f_{T-1}.$$

In other words, each $r_{t'}$ has its own row of f-value to which it gets distributed. Next, switch to the column view: instead of summing row-wise, sum column-wise. The first column is $f_0 \cdot \left(\sum_{T-1}^{t=0} r_t\right)$. The second is $f_1 \cdot \left(\sum_{T-1}^{t=1} r_t\right)$. And so on. Doing this means we get the desired formula after replacing f_t with its real meaning and hitting the expression with an expectation.

Note: It is very easy to make a typo with these. I checked my math carefully and cross-referenced it with references online (which themselves have typos). If any readers find a typo, please let me know.

Using the above formulation, we finally introduce our baseline b, which is a function of s_t (and not $s_{t'}$, I believe). We "insert" it inside the term in parentheses:

$$\nabla_\theta E_{\tau\sim\pi_\theta}[R(\tau)] = E_{\tau\sim\pi_\theta}\left[\sum_{t=0}^{T-1}\nabla_\theta \log \pi_\theta(a_t|s_t)\left(\sum_{t'=t}^{T-1} r_{t'} - b(s_t)\right)\right].$$

At first glance, it doesn't seem like this will be helpful, and one might wonder if this would cause the gradient estimate to become biased. Fortunately, it turns out that this is not a problem. This was surprising to me, because all we know is that $b(s_t)$ is a function of s_t. However, this is a bit misleading because usually we want $b(s_t)$ to be the expected return starting at time t, which means it really "depends" on the subsequent time steps. For now, though, just think of it as a function of s_t.

Understanding the Baseline

In this section, I first go over why inserting b above doesn't make our gradient estimate biased. Next, I will go over why the baseline reduces variance of the gradient estimate. These two capture the best of both worlds: staying unbiased and reducing variance. In general, any time you have an unbiased estimate and it remains so after applying a variance reduction technique, then apply that variance reduction!

First, let's show that the gradient estimate is unbiased. We see that with the baseline, we can distribute and rearrange and get:

$$\nabla_\theta E_{\tau\sim\pi_\theta}[R(\tau)] = E_{\tau\sim\pi_\theta}\left[\sum_{t=0}^{T-1}\nabla_\theta \log \pi_\theta(a_t|s_t)\left(\sum_{t'=t}^{T-1} r_{t'}\right) - \sum_{t=0}^{T-1}\nabla_\theta \log \pi_\theta(a_t|s_t)b(s_t)\right].$$

Due to linearity of expectation, all we need to show is that for any single time t, the gradient of $\log \pi_\theta(a_t|s_t)$ multiplied with $b(s_t)$ is zero. This is true because

$$\begin{aligned}
&E_{\tau\sim\pi_\theta}[\nabla_\theta \log \pi_\theta(a_t|s_t)b(s_t)] \\
&= E_{s_{0:T},a_{0:T-1}}\left[E_{s_{t+1:T},a_{t:T-1}}\left[\nabla_\theta \log \pi_\theta(a_t|s_t)b(s_t)\right]\right] \\
&= E_{s_{0:T},a_{0:T-1}}\left[b(s_t)\cdot E_{a_t}\left[\nabla_\theta \log \pi_\theta(a_t|s_t)\right]\right] \\
&= E_{s_{0:T},a_{0:T-1}}\left[b(s_t)\cdot 0\right] \\
&= 0.
\end{aligned}$$

Note I: This notation is similar to what I had before. The trajectory $s_0, a_0, \ldots, s_T$ is now represented as $s_{0:T}, a_{0:T-1}$. In addition, the expectation is split up, which is allowed. If this is confusing, think of the definition of the expectation with respect to at least two variables. We can write brackets in any appropriately enclosed location. Furthermore, we can "omit" the unnecessary variables in going from $E_{s_{t+1:T},a_{t:T-1}}$ to E_{a_t} (see expression E above). Concretely, assuming we're in discrete-land with actions in A and states in S, this is because E evaluates to:

$$E = \sum_{a_t \in A} \sum_{s_{t+1} \in S} \dots \sum_{s_T \in S} \pi_\theta(a_t|s_t) P(s_{t+1}|s_t, a_t) \dots P(s_T|s_{T-1}, a_{T-1}) \cdot (\nabla_\theta \log \pi_\theta(a_t|s_t)) \tag{2}$$

$$= \sum_{a_t \in A} \pi_\theta(a_t|s_t) \nabla_\theta \log \pi_\theta(a_t|s_t). \tag{3}$$

This is true because of the definition of expectation, whereby we get the joint density over the entire trajectory, and then we can split it up like we did earlier with the gradient of the log probability computation. We can distribute $\nabla_\theta \log \pi_\theta(a_t|s_t)$ all the way back to (but not beyond) the first sum over a_t. Pushing sums "further back" results in a bunch of sums over densities, each of which sums to one. The astute reader will notice that this is precisely what happens with variable elimination for graphical models. (The more technical reason why "pushing values back through sums" is allowed has to do with abstract algebra properties of the sum function, which is beyond the scope of this post.)

Note II: This proof above also works with an infinite-time horizon. In Appendix B of the Generalized Advantage Estimation paper (arXiv link), the authors do so with a

1 Result

The Following picture be the rewards versus episods when we do not use the baseline

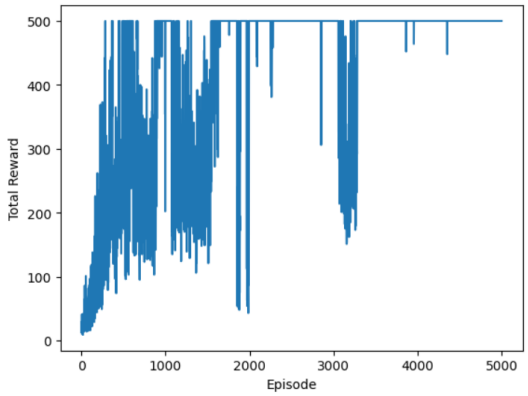

Figure 1: Rewards vs Episods

The standard policy gradient formula is adapted to incorporate discounting, which involves modifying the sum of future rewards in the gradient estimate. The discount factor is introduced to weigh future rewards, with $\gamma^{t'-t}$ representing the discount factor for each time step t' in the sum. The discounted baseline $b(s_t)$ is introduced, aiming to approximate the expected return starting at time t. The author then transitions to the concept of advantage functions, defining $Q^\pi(s, a)$ and $V^\pi(s)$ as value functions. The advantage function $A^\pi(s, a)$ is introduced as the difference between the action value and the state value. The paragraph concludes by expressing the policy gradient in terms of advantage functions, suggesting that estimating accurate advantage functions becomes crucial in policy gradient optimization.

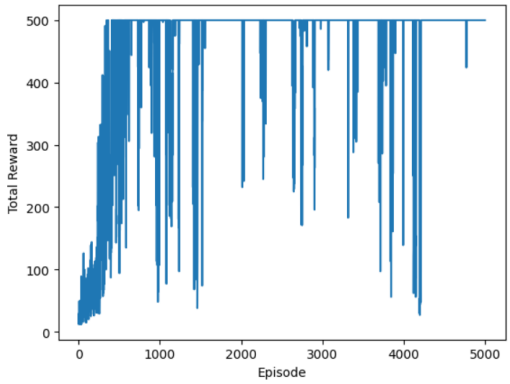

Figure 2: Rewards vs Episods

Conclusion

In conclusion, the introduction of a baseline in policy gradient optimization proves to be a valuable strategy for reducing variance. The formulation incorporates a discounted baseline, taking into account the expected return starting at a specific time, and leverages advantage functions to express the policy gradient in terms of the difference between action and state values. This innovative approach not only maintains an unbiased gradient estimate but also significantly reduces the variance. The emphasis on accurate estimation of advantage functions underscores their pivotal role in enhancing the performance of policy gradient methods. Overall, the use of a baseline emerges as an effective technique in addressing the challenges of high variance, contributing to the stability and efficiency of policy gradient optimization algorithms.