Policy Optimization with Penalized Point Probability Distance: an Alternative to Proximal Policy Optimization

Xiangxiang Chu
Xiaomi AI Lab
chuxiangxiang@xiaomi.com

Abstract

As the most influential variant and improvement for Trust Region Policy Optimization (TRPO), proximal policy optimization (PPO) has been widely applied across various domains with its inherent advantages involving sample efficiency, implementation and parallelism after published. In this paper, a first order gradient reinforcement learning algorithm called policy optimization with penalized point probability distance (POP3D) is proposed as another variant for TRPO and the point probability distance is proven to be a lower bound of the total variance divergence while has inherent advantage over exploration. The paper is organized as follows. First, we start a discussion about weakness of several commonly used algorithms, from which our method are motivated. Secondly, we propose our algorithm to overcome these shortcomings. Then we make more explanations about PPO’s improvement mechanism over TRPO from the perspective of solution manifold. Finally, we make quantitative comparisons among several state-of-the-art algorithms based on OpenAI Atari and Mujoco environments, where a baseline is specially designed to act as ablation for our improvement. While keeping almost all beneficial spheres of PPO, it encourages more exploration and overcomes the shortcoming of using Kullback-Leibler divergence, which is prone to instability. Simulation results show that POP3D is highly competitive compared with PPO, since it reaches state-of-the-art within 40 million frame steps on 49 Atari games and competitive scores in continuous domain according to common metrics: final performance and fast learning ability.

1. Introduction

With combination of deep learning with reinforcement approach, lots of impressive results have been produced in a wide range of fields, from playing Atari game[12, 6], controlling robotics[10], Go[21] to neural architecture search[23, 14]. In the case of Atari game and Go, agent receives raw pictures as inputs and makes decisions depending on automatically learned features by deep neural networks.

The basis of reinforcement learning algorithm is generalized policy iteration[22], which states the two essential iterative steps: policy evaluation and improvement. Among various algorithms, policy gradient is an active branch of reinforcement learning whose foundation is Policy Gradient Theorem and most classical algorithm REINFORCEMENT[22], can date back last centuries. Afterwards, with the rapid growth of deep learning, handfuls of policy gradient based algorithms are proposed, such as Deep Deterministic Policy Gradient (DDPG)[10], Asynchronous Advantage Actor Critic(A3C)[13], Actor Critic using Kronecker-factored Trust Region(ACKTR)[27], Proximal Policy Optimization(PPO)[20] and so on. Improving the strategy monotonically has been nontrivial until the trust region policy optimization(TRPO) is proposed[18]. Hessian free strategy - Fisher vector product is used to cut down the computing burden. Kullback-Leibler divergence(KLD, also called relative entropy) is taken from the objective and acts as a hard constraint since its corresponding coefficient is difficult to tune and vary greatly for different problems. Moreover, TRPO still have several drawbacks when applied in practice. It’s too complicated and needs to perform several Fisher vector product within one iteration. Quite a lot of efforts have been devoted to improve TRPO since then and the most distinguished one is PPO.

Schulman et al. propose PPO, which be regraded as a first-order variant of TRPO but have great improvements in several facets[20]. A pessimistic clipped surrogate objective is proposed, where TRPO’s hard constraint is imposed implicitly by clipped action probability ratio. Since its objective has been turned into solving an unconstrained optimization problem, any first-order stochastic gradient can be applied directly. It’s easier to implement, more robust to various problems and achieve state-of-the-art result on Atari games[2].
Therefore, it’s widely used since invented\(^1\) [15, 28] and even becomes OpenAI’s default reinforcement learning algorithm.

As another potential improvement for TRPO and an alternative to PPO, this paper focuses on a policy optimization algorithm, where its contributions are:

1. It offers a simple variant of TRPO called POP3D along with a new surrogate objective containing a point probability penalty item, which is a lower symmetric bound to the total variance divergence for two policies. Moreover, it helps to stabilize learning process and encourage exploration. Furthermore, it escapes from penalty item setting headache along with TRPO, where is arduous to select one fixed value for various environments.

2. It achieves state-of-the-art results with clear margin on 49 Atari games within 40 million frame steps based on two shared metrics. Moreover, it also achieves competitive results compared with PPO in continuous domain.

3. It dives into the mechanism of PPO’s improvement over TRPO by the perspective of solution manifold.

4. It enjoys almost all PPO’s advantages such as easy implementation, fast learning ability, which is an alternative to PPO.

This paper is organized as follows. First, we introduce essential background knowledge for better understanding of the following sections. Second, we discuss about weakness of several commonly used algorithms. Specially, applying KLD which is an asymmetric upper bound to square of total variance divergence not only has the modal match drawback, but imposes a strict limitation on the solution manifold. Then we propose our algorithm to overcome these corresponding shortcomings and analyze one of PPO’s active improvement mechanism over TRPO from the perspective of manifold, which is also owned by POP3D. Last, we evaluate our method thoroughly on Atari 49 games and Mujoco 7 games on widely used metrics using different seeds.

2. Background and Related Works

2.1. Policy Gradient

Agents interact with environment and receive reward which is used to adjust their policy in turn. At state \(s_t\), one agent takes strategy \(\pi\) and goes to a new state \(s_{t+1}\), rewarded \(r_t\) by the environment. Maximizing discounted return(accumulated rewards) \(R_t\) is the agent’s objective. In specific, given a policy \(\pi\), \(R_t\) is defined as

\[
R_t = \sum_{n=0}^{\infty} (r_t + \gamma r_{t+1} + \gamma^2 r_{t+2} + \ldots + \gamma^n r_{t+n}).
\]  

\(\gamma\) is the discounted coefficient to control future rewards, which is in the range from 0 to 1. Regarding a neural network with parameter \(\theta\), the policy \(\pi_\theta(a|s)\) can be learned by maximizing Equation 1 using back propagation algorithm. In specific, with \(Q(s, a)\) representing the agent’s return in state \(s\) after taking action \(a\), the objective function, which is well-known as policy gradient theorem, can be written as

\[
\max_\theta \quad E_{s, a} \log \pi_\theta(a|s)Q(s, a).
\]  

Equation 2 lays foundation for handfuls of policy gradient based algorithms, such as REINFORCEMENT, actor critic[9] and A3C. Another variant can be deduced by using

\[
A(s, a) = Q(s, a) - V(s)
\]  

to replace \(Q(s, a)\) in Equation 2 equivalently, \(V(s)\) can be any function as longs as \(V\) depends on \(s\) but not \(a\). In most cases, \(V\) is state value function, which not only helps to reduce variations but has clear physical meaning. Therefore, the variant can be written as

\[
\max_\theta \quad E_{s, a} \log \pi_\theta(a|s)A(s, a).
\]

2.2. Advantage Estimate

One commonly used method for advantage calculation is one step estimation, which estimates

\[
A(s_t, a) = Q(s_t, a) - V(s_t)
= r_t + \gamma V(s_{t+1}) - V(s_t).
\]  

A better estimate for advantage called generalized advantage estimation is proposed in [19], where one, two, three, up to \(\infty\) time step estimate is combined and summarized using \(\lambda\) based weights, which is beneficial to estimate more accurately. The generalized advantage estimator is defined as

\[
\hat{A}^\text{GAE}(\gamma, \lambda) = \sum_{l=0}^{\infty} (\gamma\lambda)^l \delta_{t+l}^V
\]

\[
\delta_{t+l}^V = r_{t+l} + \gamma V(s_{t+l+1}) - V(s_{t+l}).
\]

The parameter \(\lambda\) meets \(0 \leq \lambda \leq 1\), which controls the trade-off between bias and variance.

2.3. Trust Region Policy Optimization

In TRPO[18], an objective with penalized KL divergence is derived to monotonically update the policy.
Specifically,

$$\max_{\theta} \ E_t \left[ \pi_\theta(a_t | s_t) - C \sum_{i=1}^{D} KL[\pi_{\theta_{old}}(\cdot | s_t) || \pi_\theta(\cdot | s_t)] \right]$$

$$\quad - CE_t[KL[\pi_{\theta_{old}}(\cdot | s_t), \pi_\theta(\cdot | s_t)]]$$

$$C = \frac{2c\gamma}{(1-\gamma)^2}$$

$$\epsilon = \max_s E_{a \sim \pi_\theta(a | s)}[A_{\pi_{\theta_{old}}}(s, a)]$$

where $C$ is the penalty coefficient, which can be derived as $[18]$. In practice, the policy update steps would be too small if $C$ is valued as derived. Moreover, it’s impossible to calculate $C$ accurately beforehand since it requires traverse all states to reach the maximum and inevitable bias as well as variance introduced by estimating the advantages for old policy while training. Instead, a surrogate objective is maximized based on the KLD constraint between the old and new policy, which can be written as below,

$$\max_{\theta} \ E_t \left[ \pi_\theta(a_t | s_t) - C \sum_{i=1}^{D} KL[\pi_{\theta_{old}}(\cdot | s_t) || \pi_\theta(\cdot | s_t)] \right]$$

$$s.t. \ E_t[KL[\pi_{\theta_{old}}(\cdot | s_t), \pi_\theta(\cdot | s_t)]] \leq \delta$$

where $\delta$ is the KLD upper limitation. In addition, conjugate gradient is applied to solve Equation 8 more efficiently. Two major problems have yet to be addressed: one is its complexity even using conjugate gradient, another is not compatible with architectures that include noise or parameter sharing$[20]$.

### 2.4. Proximal Policy Optimization

PPO can be regarded as a variant of TRPO. It replaces the original constrained problem with a pessimistic clipped surrogate objective where KL constraint is implicitly imposed. PPO uses several actors to collect the data and performs policy update $K$ times based on the sample data. The loss function can be written as

$$L^{CLIP}_{\text{PPO}}(\theta) = E_t \left[ \min(r_t(\theta), 1 - \epsilon, 1 + \epsilon) A_t \right]$$

where $\epsilon$ is a hyper-parameter to control the clipping ratio. Except for the clipped PPO version, KL penalty versions including fixed and adaptive KLD are used as baselines for comparison with clipped PPO. And the simulation results convince that clipped PPO is the best solution for various domain.

Recently, an adaptive clipping approach for PPO(PPO-$\lambda$) is proposed by $[3]$ base on PPO may fail to adaptively improve learning performance in accordance with the importance of each sampled state. However, only quite a few games are used to compare PPO-$\lambda$ and PPO, and several experiment settings such as game frames are not the same as $[20]$, which may lead to bias.

### 3. Policy Optimization with Penalized Point Probability Distance

Before diving into our algorithm POP3D, firstly we review some potential weakness exposed by several baselines, which partly motivate us to propose our methods.

#### 3.1. Disadvantages of Kullback-Leibler Divergence

Let’s start our journey with a quick review of the birth of TRPO. In general, the constrained trust region form of TRPO is derived from a surrogate objective containing the KLD penalty as Equation 7. Owing to the intractability mentioned section 2.3 for setting $C$, Schulman et al. approximately transform unconstrained optimization into constrained problem with KL constraint(Equation 8).

According to Theorem 1 in TRPO paper, the following inequality holds$[18]$.

$$\eta(\pi_\theta) \leq L_{\pi_{\theta_{old}}}(\pi_\theta) + \frac{2c\gamma}{(1-\gamma)^2} \alpha^2$$

$$\alpha = D^{\max}_{TV}(\pi_{\theta_{old}}, \pi_{\theta})$$

$$D^{\max}_{TV}(\pi_{\theta_{old}}, \pi_{\theta}) = \max_s D_{TV}(\pi_{\theta_{old}} || \pi_{\theta})$$

Equation 10 from $[18]$ replaces total variation divergence $D^{\max}_{TV}(\pi_{\theta_{old}}, \pi_{\theta})$ by $D^{\max}_{KL}(\pi_{\theta_{old}}, \pi_{\theta}) = \max_s D_{KL}(\pi_{\theta_{old}} || \pi_{\theta})$. Whereas, for discrete distribution $p$ and $q$, their total variation divergence $D_{TV}(p||q)$ is defined by $\frac{1}{2} \sum_i |p_i - q_i|$. Obviously, $D_{TV}$ is symmetric by definition, while KLD is an asymmetric distance measurement for two probability distributions with inherent weakness.

In discrete domain, given state $s$, KLD of $\pi_{\theta_{old}}(\cdot | s)$ for $\pi_{\theta}(\cdot | s)$ can be written as

$$D_{KL}(\pi_{\theta_{old}}(\cdot | s || \pi_{\theta}(\cdot | s)) = \sum_a \pi_{\theta_{old}}(a | s) \ln \frac{\pi_{\theta}(a | s)}{\pi_{\theta_{old}}(a | s)}$$

Similarly, KLD in continuous domain can be defined simply by replace summation with integration.

The consequence of KLD’s asymmetry leads to non-negligible difference of whether choose $D_{KL}(\pi_{\theta_{old}} || \pi_{\theta})$ or $D_{KL}(\pi_{\theta} || \pi_{\theta_{old}})$. Sometimes, those two choices result in quite different solutions. Robert compared the forward and reverse KL on a distribution, one solution matches only one of the modes, and another covers both modes$[16]$.

In a word, KLD is not an ideal bound or approximation for the expected discounted cost.
3.2. Discussion on Pessimistic Proximal Policy

As a matter of fact, PPO should be called pessimistic proximal policy optimization because of the loss function is composed of minimum of \( r_1(\theta)A_t \) and \( \text{clip}(r_1(\theta), 1 - \epsilon, 1 + \epsilon)A_t \).

Without loss of generality, assuming \( A_t > 0 \) for given state \( s_t \) and action \( a_t \), and the optimal choice is \( a^*_t \). When \( a_t = a^*_t \), a good update policy is to increase the probability of action to a relative high value \( a^*_t \) by adjusting \( \theta \). However, the clipped item \( \text{clip}(r_1(\theta), 1 - \epsilon, 1 + \epsilon)A_t \) will fully contribute to the loss function by the minimum operation. Other situation with \( A_t < 0 \) can be analyzed in the same manner.

If the pessimistic limitation is removed, its performance decreases dramatically\([20]\), which is again confirmed by our preliminary experiments. In other words, the pessimistic mechanism plays a very essential role for PPO by relatively weak preference for good action decision for a given state, which in turn affects learning efficiency.

3.3. Restricted Solution Manifold

Suppose \( \pi_{\theta^*} \) is the optimal solution for a given environment, in most cases, more than one parameter setting for \( \theta \) can generate the ideal policy, especially when \( \pi_{\theta^*} \) is learned by deep neural network. In other words, the relationship between \( \theta \) and \( \pi^* \) is many to one. On the other hand, when agents interact with environment using policy represented by neural networks, the action is taken approximately strongly correctly with the highest probability value. Although some strategies of enhancing exploration are applied, they don’t affect the policy much in the meaning of expectation.

Taking the Atari-Pong game for example, when an agent sees a Pong ball is coming nearly, its optimal policy is move the racket to the right position. The probability of this action is a relatively high value such as 0.95 and it’s near to impossible that this value is 1.0 since it’s produced by a softmax operation on the several discrete actions. In fact, we hardly obtained the optimal solution accurately, instead, our goal is a good enough answer. Namely, \( \theta_1 \) outputting a probability 0.95 and \( \theta_2 \) with 0.9 for the right action are both good answers. During the training process, this similar events occur frequently.

Using a penalty such as KLD cannot handle it effectively, because it involves all of actions’ probabilities. Moreover, it doesn’t stop penalizing unless two distributions become exactly indifferent. Therefore, even if \( \theta \) outputs \( \theta_{\text{old}} \) the same high probability for the right action, it’s still penalized owing to probabilities mismatch for other uncritical actions. Indeed, when a person is asked to make the choice, corresponding action will be taken only if the probability is above a threshold. From the perspective of manifold, if the optimal parameters constitute a solution manifold. The KLD penalty will act until \( \theta \) exactly locates in the solution if possible. However, if the agent concentrates only on critical actions like human, it’s much easier to approach the manifold, which in fact expands the solution manifold at least one dimension such as curves to surfaces and surfaces to spheres.

A simple example helps throw light on this insight. Suppose the target probability for a given state with ten action choice is 0.847 for the target action, and 0.017 for every other actions. The parameter update process can be simulated by 2000 subtle perturbations around the target distribution by three trials and calculate statistic features, which is summarized in Table 1. On top of that, we can imagine that a single point is expanded by a line segment.

Besides, since min-batch is a commonly used tricks for training neural networks, removing this unexpected penalties has the effect of decreasing penalty noises, which are reflected by corresponding gradient.

3.4. Exploration

One shared highlight in reinforcement learning is the balance between exploitation and exploration. For policy gradient based algorithm, entropy is added in total loss to encourage exploration in most case. When included in loss function, KLD penalizes the old and new policy probability mismatch for all possible actions as Equation 11. This strict punishment for every action’s probability mismatch, which discourages exploration.

3.5. Point Probability Distance

To overcome the above shortcomings, we propose a surrogate objective with the point probability distance penalty, which is symmetric and less pessimistic than PPO. In discrete domain, when the agent takes action \( a \), then point probability distance between \( \pi_{\theta_{\text{old}}} (\cdot | s) \) and \( \pi_{\theta} (\cdot | s) \) is defined by

\[
D_{pp}(\pi_{\theta_{\text{old}}} (\cdot | s), \pi_{\theta} (\cdot | s)) = (\pi_{\theta_{\text{old}}} (a | s) - \pi_{\theta} (a | s))^2
\]

Some attentions should be paid to the penalty definition item, the distance is measured by the point probability, which emphasizes its mismatch for the sampled actions for corresponding state. Undoubtedly, \( D_{pp} \) is symmetric by definition. Furthermore, it can be proved that \( D_{pp} \) is indeed

| Experiments | \( D_{KL_{\text{min}}} \) | \( D_{KL_{\text{max}}} \) | \( D_{KL_{\text{mean}}} \) | \( D_{KL_{\text{std}}} \) |
|-------------|-----------------|-----------------|-----------------|-----------------|
| 1           | 0.0003          | 0.0423          | 0.0115          | 0.0067          |
| 2           | 0.0010          | 0.0447          | 0.0114          | 0.0065          |
| 3           | 0.0007          | 0.0422          | 0.0114          | 0.0068          |

Table 1. Simple perturbation experiments for KLD.

a symmetric lower bound for the total variance divergence $D_{TV}$. In the meanwhile, it can be interpreted as a good approximation about $D_{TV}$ under some conditions while contributes to approaching solution manifold more easily. As a special case, it can be easily proved that for a binary distribution, $D_{TV}(p||q) = D_{pp}(p||q)$.

**Theorem 1** For two discrete probability distributions $p$ and $q$ with $K$ values, then $D_{TV}(p||q) \geq D_{pp}(p||q)$ holds.

**Proof 1** Let $a = p_i$, $b = q_i$ for any $l$, and suppose $a \geq b$ without loss of generalization. So,

$$D_{TV}(p||q) = \frac{1}{2} \sum_{i=1}^{K} |p_i - q_i|$$

$$\geq \frac{1}{2} \sum_{i=1, i \neq l}^{K} p_i - q_i| + \frac{1}{2} |p_l - q_l|$$

$$\geq \frac{1}{2} \sum_{i=1, i \neq l}^{K} p_i - q_i| + \frac{1}{2} (a - b)$$

$$\geq \frac{1}{2} (1 - a - (1 - b)) + \frac{1}{2} (a - b)$$

$$\geq \frac{1}{2} (a - b) + \frac{1}{2} (a - b)$$

$$= D_{pp}(p||q)$$

Since $0 \leq \pi_{\theta}(a|s) \leq 1$ holds for discrete action space, $D_{pp}$ has lower and upper boundary: $0 \leq D_{pp} \leq 1$. Moreover, $D_{pp}$ is less sensitive to action space dimension than KLD, which has a similar effect as PPO’s clipped ratio to increase robustness and enhance stability.

Equation 12 stays unchanged for continuous domain, and the only difference is $\pi_{\theta}(a|s)$ represents point probability density instead of probability.

### 3.6. POP3D

After we have defined the point probability distance, we use a new surrogate objective for POP3D, which can be written as

$$\max_{\theta} \quad E_{\pi_{\theta}}[\frac{\pi_{\theta}(a_t|s_t)}{\pi_{\text{old}}(a_t|s_t)} \hat{A}_t - \beta D_{pp}(\pi_{\text{old}}(\cdot|s_t), \pi_{\theta}(\cdot|s_t))],$$

where $\beta$ is the penalized coefficient. These combined advantages lead to considerable performance improvement, which escapes from the dilemma for penalty coefficient setting. In implementation, we use generalized advantage estimates to calculate $\hat{A}_t$. Algorithm 1 shows the complete iteration process of POP3D. Moreover, it possesses the same computational cost and data efficiency as PPO.

| Algorithm 1 POP3D |
|---------------------------------------------|
| **Input:** max iterations $L$, actors $N$, epochs $K$ |
| **for iteration** $= 1$ to $L$ **do** |
| **for actor** $= 1$ to $N$ **do** |
| Run policy $\pi_{\text{old}}$ for $T$ time steps |
| Compute advantage estimations $\hat{A}_1, ..., \hat{A}_T$ |
| **end for** |
| **for epoch** $= 1$ to $K$ **do** |
| Optimized loss objective wrt $\theta$ with mini-batch size $M \leq NT$, then update $\theta_{\text{old}} \leftarrow \theta$. |
| **end for** |
| **end for** |

### 3.7. Relationship With PPO

To conclude this section, we take some time to see why PPO works by taking the above viewpoints into account.

When we pour more attention to Equation 9, the ratio $r_t(\theta)$ only involves the probability for given action $a$, which is chosen by policy $\pi$. In other words, all other actions’ probabilities except $a$ are not activated, which no longer contribute to back propagation and allow probability mismatch. Obviously, this procedure behaves similarly as POP3D, which expands the restricted solution manifold.

Above all, POP3D is designed to conform with the regulations for overcoming above mentioned problems, and in the next section experiments from commonly used benchmarks will evaluate its performance.

## 4. Experiments

### 4.1. Controlled Experiments Setup

This section is concerned with controlled experiments setting.

OpenAI Gym is a distinguished simulation environment to test and evaluate various reinforcement algorithms, which is composed of both discrete(Atari) and continuous(Mujoco) domains [2]. Most of deep reinforcement learning methods within recent three years such as DQN variants [24, 25, 17, 1, 6], A3C[11], ACKTR[26] and PPO[20] are evaluated using only one set of hyper-parameters². Therefore, we evaluate POP3D’s performance on 49 Atari games(version-4, discrete action space) and 7 Mujoco(version-2, continuous space).

Since PPO is state of the art RL algorithm which beats various methods such as A3C, A2C³ and ACKTR, we focus on a detailed quantitative comparison with fine-tuned PPO. And we don’t consider large scale distributed algorithms

²DQN variants are evaluated in Atari environment since they are designed to solve problems about discrete action space. However, policy gradient based algorithms can handle both continuous and discrete problems.

³A2C can be regarded as a synchronous version to A3C.
Apex-DQN[7] and IMPALA[4], because we concentrate on comparable and fair evaluation, while the latter is designed to apply with large scale parallelism. Nevertheless, some orthogonal tricks from those methods have the potentials to improve our method further. In our opinion, it’s not a sensible choice if we make comparison at the cost of justice loss. Furthermore, we include TRPO acts as a baseline method to compare since our algorithm can be regarded as its variant. In addition, quantitative comparisons between KLD and point probability penalty helps to verify the critical role of the latter, where the former strategy is named fixed KLD in [20] and can act as another good baseline in this context, named by BASELINE below.

In specific, we retrained one agent for each game with fine-tuned hyper-parameters. To avoid the problems of reproduction about reinforcement algorithms mentioned in [5], we take the following measures:

- Use the same training steps and make use of the same amount of game frames (40M for Atari game and 10M for Mujoco).
- Use the same neural network structures, which is CNN model with one action head and one value head for Atari game, and fully-connected model with one value head and one action head which produce the mean and std parameter of diagonal Gaussian distribution as [20].
- Initialize parameters using the same strategy as [20].
- Keep Deepmind Gym wrappers such as reward clipping and frame stacking unchanged for Atari domain, and enable 30 no-ops at the beginning of each episode.
- Use Adam optimizer [8] and decrease $\alpha$ linearly from 1 to 0 for Atari domain as [20].

To facilitate further comparisons with other approaches, we release the seeds and detailed results (across the entire training process for different trials). In addition, we randomly select three seeds from \{0, 10, 100, 1000, 10000\} for two domains, \{10,100,1000\} for Atari and \{0,10,100\} for Mujoco in order to decrease unfavorable subjective bias stated in [5].

4.2. Evaluation Metrics

Two score metrics for evaluating agents performance using various RL algorithms is presented by Schulman et al. [20]. One is mean score of last 100 episodes $\text{Score}_{100}$, which measures how high a strategy can hit eventually. Another is average score across all episodes $\text{Score}_{\text{all}}$, which measure how fast an agent learns. In this paper, we conform to this routine and calculate individual metric by averaging three trials in the same way.

4.3. Discrete Domain Comparisons

4.3.1 Hyper-parameters

We perform four hyper-parameter searches for the penalty coefficient $\beta$ based on four Atari games while keeping other hyper-parameters unchanged from PPO and fix $\beta = 5.0$ to train all Atari games. For BASELINE, we also perform four hyper-parameter searches on penalty coefficient $\beta$ and choose $\beta = 10.0$. To save space, detailed hyper-parameter setting can be found in Table 6, 7 and 8.

Apparently this process is not beneficial for POP3D owing to missing optimization for all hyper-parameters. Nevertheless, there are several reasons behind that choice. On the one hand, it’s the easiest option to make a relative fair comparison group such as keeping the same total iteration number and epochs within one loop to our knowledge. On the other hand, this process imposes low search requirements for time and resource. That’s to say, conservative conclusions can be drawn that our method is better than, at least competitive to PPO if it still behaves better on common benchmarks.

4.3.2 Comparison Results

The final score of each game is averaged by three trials and the highest is in bold. As Table 2 shows, POP3D wins with obvious margin: 32 out of 49 Atari games in view of final score, followed by PPO with 11, BASELINE with 5 and TRPO with 1. Interestingly, for games that POP3D score highest, BASELINE score worse than PPO more often than the other way round, which means that POP3D is not just an approximate version of BASELINE.

For another metric, POP3D wins 20 out of 49 Atari games which matches PPO with 18, followed by BASELINE with 6, and last ranked by TRPO with 5.

Detailed performance during the whole training process can be found in Figure 1. If we measure the stability of an algorithm by score variance of different trials, POP3D score high with good stability across various seeds. And PPO shows relatively worse stability in Game Kangaroo and UpNDown. Interestingly, BASELINE shows large variance for different seeds for several games such as BattleZone, Freeway, Pitfall and Seaquest.
In sum, POP3D reveals its better capacity to score high and similar fast learning ability in this domain. Detailed metric for each game is listed in Table 11 and 12.

| Metric | PPO | POP3D | BASELINE | TRPO |
|--------|-----|-------|----------|------|
| Score100 | 11  | 32    | 5        | 1    |
| Scoreall | 18  | 20    | 6        | 5    |

Table 2. Number of games "won" by each algorithm for Atari game, where the score metric is averaged on three seeds.

### 4.4. Continuous Domain Comparisons

In this section, we focus on comparisons between POP3D and PPO in Mujoco domain.

#### 4.4.1 Hyper-parameters

For PPO, we use the same hyper-parameter configuration as [20]. Regarding POP3D, we perform 3 searches on two games and select 5.0 as the penalty coefficient. More details about hyper-parameters for PPO and POP3D can be referred to Table 9 and 10 of Section A.2. Unlike the Atari domain, we use constant learning rate as [20] in continuous domain instead of linear decrease.

#### 4.4.2 Comparison Results

The scores are also averaged on three trials and summarized in Table 3. POP3D occupies 5 out of 7 games on Score100, and 3 on Scoreall. Evaluation metrics of both methods across different games are illustrated in Table 4 and 5. Each algorithm’s score performance with iteration steps is shown in Figure 2.

In summary, both metrics indicates that POP3D is competitive to PPO in continuous domain.

| game                  | PPO | POP3D |
|-----------------------|-----|-------|
| HalfCheetah           | 2726.03 | 3184.54 |
| Hopper                | 2027.21 | 1452.09 |
| InvertedDoublePendulum| 4455.03 | 4907.64 |
| InvertedPendulum      | 544.02  | 741.94  |
| Reacher               | -5.00  | -4.29  |
| Swimmer               | 111.88  | 111.08  |
| Walker2d              | 1112.25 | 3966.01 |

Table 3. Mean final scores (last 100 episodes) of PPO ,POP3D on Mujoco games after 10M frames. The results are averaged on three trials.

| game                  | PPO | POP3D |
|-----------------------|-----|-------|
| HalfCheetah           | 3250.22 | 2373.30 |
| Hopper                | 1767.14 | 1257.72 |
| InvertedDoublePendulum| 3684.92 | 2561.77 |
| InvertedPendulum      | 531.77  | 552.98  |
| Reacher               | -5.94  | -8.05  |
| Swimmer               | 94.01  | 108.27  |
| Walker2d              | 1770.37 | 2439.54 |

Table 4. All episodes mean scores of PPO ,POP3D on Mujoco games after 10M frames. The results are averaged by three trials.

In summary, both metrics indicates that POP3D is competitive to PPO in continuous domain.

#### 5. Conclusion

In this paper, we introduce a new reinforcement learning algorithm called POP3D(policy optimization with penalized point probability distance) and regard it as a TRPO variant like PPO. Compared with KLD that is an upper bound for the total variance divergence, the penalized point probability distance is indeed a symmetric lower bound. Besides, it equivalently expands the optimal solution manifold effectively while encourage exploration, which is a similar mechanism implicitly possessed by PPO. The proposed method not only inherits several critical improvements from PPO, but outperforms with obvious margin on Atari 49 games in view of final scores and meets PPO’s match as for fast learning ability.

More interestingly, it not only hardly suffers from penalty item setting headache along with TRPO, where is arduous to select one fixed value for various environments, but outperform fixed KLD baseline from PPO. In summary, POP3D is highly competitive and an alternative to PPO.

#### References

[1] M. G. Bellemare, W. Dabney, and R. Munos. A distributional perspective on reinforcement learning. *arXiv preprint arXiv:1707.06887*, 2017.

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

[3] G. Chen, Y. Peng, and M. Zhang. An adaptive clipping approach for proximal policy optimization. *arXiv preprint arXiv:1804.06461*, 2018.

[4] L. Espeholt, H. Soyer, R. Munos, K. Simonyan, V. Mnih, T. Ward, Y. Doron, V. Firouz, T. Harley, I. Dunning, et al. Impala: Scalable distributed deep-rl with importance weighted actor-learner architectures. *arXiv preprint arXiv:1802.01561*, 2018.
Figure 1. Score curves of three methods on 49 Atari games within 40 million frame steps.
Figure 2. Score curves of three methods on 7 Mujoco games within 10 million frame steps.

| Method | Score | Frames/M |
|--------|-------|----------|
| PPO    |       |          |
| PPO    |       |          |
| PPO    |       |          |

[5] P. Henderson, R. Islam, P. Bachman, J. Pineau, D. Precup, and D. Meger. Deep reinforcement learning that matters. *arXiv preprint arXiv:1709.06560*, 2017.

[6] M. Hessel, J. Modayil, H. Van Hasselt, T. Schaul, G. Ostrovski, W. Dabney, D. Horgan, B. Piot, M. Azar, and D. Silver. Rainbow: Combining improvements in deep reinforcement learning. *arXiv preprint arXiv:1710.02298*, 2017.

[7] D. Horgan, J. Quan, D. Budden, G. Barth-Maron, M. Hessel, H. Van Hasselt, and D. Silver. Distributed prioritized experience replay. *arXiv preprint arXiv:1803.00933*, 2018.

[8] D. P. Kingma and J. Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[9] V. R. Konda and J. N. Tsitsiklis. Onactor-critic algorithms. *SIAM journal on Control and Optimization*, 42(4):1143–1166, 2003.

[10] T. P. Lillicrap, J. J. Hunt, A. Pritzel, N. Heess, T. Erez, Y. Tassa, D. Silver, and D. Wierstra. Continuous control with deep reinforcement learning. *arXiv preprint arXiv:1509.02971*, 2015.

[11] V. Mnih, A. P. Badia, M. Mirza, A. Graves, T. Lillicrap, T. Harley, D. Silver, and K. Kavukcuoglu. Asynchronous methods for deep reinforcement learning. In *International Conference on Machine Learning*, pages 1928–1937, 2016.

[12] V. Mnih, K. Kavukcuoglu, D. Silver, A. A. Rusu, J. Veness, M. G. Bellemare, A. Graves, M. Riedmiller, A. K. Fidjeland, G. Ostrovski, et al. Human-level control through deep reinforcement learning. *Nature*, 518(7540):529, 2015.

[13] V. Mnih, A. Puigdomènec Badia, M. Mirza, A. Graves, T. P. Lillicrap, T. Harley, D. Silver, and K. Kavukcuoglu. Asynchronous Methods for Deep Reinforcement Learning. *ArXiv e-prints*, Feb. 2016.

[14] H. Pham, M. Y. Guan, B. Zoph, Q. V. Le, and J. Dean. Efficient neural architecture search via parameter sharing. *arXiv preprint arXiv:1802.03268*, 2018.

[15] P. Ramachandran, B. Zoph, and Q. V. Le. Searching for activation functions. 2018.

[16] C. Robert. Machine learning, a probabilistic perspective, 2014.

[17] T. Schaul, J. Quan, I. Antonoglou, and D. Silver. Prioritized experience replay. *arXiv preprint arXiv:1511.05952*, 2015.

[18] J. Schulman, S. Levine, P. Moritz, M. I. Jordan, and P. Abbeel. Trust Region Policy Optimization. *ArXiv e-prints*, Feb. 2015.

[19] J. Schulman, P. Moritz, S. Levine, M. Jordan, and P. Abbeel. High-Dimensional Continuous Control Using Generalized Advantage Estimation. *ArXiv e-prints*, June 2015.

[20] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov. Proximal Policy Optimization Algorithms. *ArXiv e-prints*, July 2017.

[21] D. Silver, J. Schrittwieser, K. Simonyan, I. Antonoglou, A. Huang, A. Guez, T. Hubert, L. Baker, M. Lai, A. Bolton, et al. Mastering the game of go without human knowledge. *Nature*, 550(7676):354, 2017.

[22] R. S. Sutton and A. G. Barto. *Reinforcement learning: An introduction*, volume 1. MIT press Cambridge, 1998.

[23] M. Tan, B. Chen, R. Pang, V. Vasudevan, and Q. V. Le. Mnasnet: Platform-aware neural architecture search for mobile. *arXiv preprint arXiv:1807.11626*, 2018.

[24] H. Van Hasselt, A. Guez, and D. Silver. Deep reinforcement learning with double q-learning. In *AAAI*, volume 16, pages 2094–2100, 2016.

[25] Z. Wang, T. Schaul, M. Hessel, H. Van Hasselt, M. Lanctot, and N. De Freitas. Dueling network architectures for deep reinforcement learning. *arXiv preprint arXiv:1511.06581*, 2015.

[26] Y. Wu, E. Mansimov, R. B. Grosse, S. Liao, and J. Ba. Scalable trust-region method for deep reinforcement learning using kronecker-factored approximation. In *Advances in neural information processing systems*, pages 5279–5288, 2017.

[27] Y. Wu, E. Mansimov, S. Liao, R. Grosse, and J. Ba. Scalable trust-region method for deep reinforcement learning using Kronecker-factored approximation. *ArXiv e-prints*, Aug. 2017.

[28] B. Zoph, V. Vasudevan, J. Shlens, and Q. V. Le. Learning transferable architectures for scalable image recognition. *arXiv preprint arXiv:1707.07012*, 2017.
A. Hyper-parameters

A.1. Atari

PPO’s and POP3D’s hyper-parameters for Mujoco game are respectively listed in Table 6 and 7.

| Hyper-parameter          | Value        |
|--------------------------|--------------|
| Horizon (T)              | 128          |
| Adam step-size           | $2.5 \times 10^{-4} \times \alpha$ |
| Num epochs               | 3            |
| Mini-batch size          | $32 \times 8$ |
| Discount ($\gamma$)      | 0.99         |
| GAE parameter ($\lambda$)| 0.95         |
| Number of actors         | 8            |
| Clipping parameter       | $0.1 \times \alpha$ |
| VF coeff.                | 1            |
| Entropy coeff.           | 0.01         |

Table 6. PPO’s hyper-parameters for Atari game.

| Hyper-parameter          | Value        |
|--------------------------|--------------|
| Horizon (T)              | 128          |
| Adam step-size           | $2.5 \times 10^{-4} \times \alpha$ |
| Num epochs               | 3            |
| Mini-batch size          | $32 \times 8$ |
| Discount ($\gamma$)      | 0.99         |
| GAE parameter ($\lambda$)| 0.95         |
| Number of actors         | 8            |
| Clipping parameter       | $0.1 \times \alpha$ |
| VF coeff.                | 1            |
| Entropy coeff.           | 0.01         |
| KL penalty coeff.        | 5.0          |

Table 7. POP3D’s hyper-parameters for Atari game.

A.2. Mujoco

PPO’s and POP3D’s hyper-parameters for Mujoco game are respectively listed in Table 9 and 10.

| Hyper-parameter          | Value        |
|--------------------------|--------------|
| Horizon (T)              | 2048         |
| Adam step-size           | $3 \times 10^{-4}$ |
| Num epochs               | 10           |
| Mini-batch size          | 64           |
| Discount ($\gamma$)      | 0.99         |
| GAE parameter ($\lambda$)| 0.95         |
| Clipping parameter       | 0.2          |

Table 9. PPO’s hyper-parameters for Mujoco game.

| Hyper-parameter          | Value        |
|--------------------------|--------------|
| Horizon (T)              | 2048         |
| Adam step-size           | $3 \times 10^{-4}$ |
| Num epochs               | 10           |
| Mini-batch size          | 64           |
| Discount ($\gamma$)      | 0.99         |
| GAE parameter ($\lambda$)| 0.95         |
| KL penalty coeff.        | 5.0          |

Table 10. POP3D’s hyper-parameters for Mujoco game.

B. Score Tables and Curves
| game             | POP3D   | PPO     | BASELINE | TPRO    |
|------------------|---------|---------|----------|---------|
| Alien            | 1510.80 | 1431.17 | 1311.23  | 1110.40 |
| Amidar           | 729.15  | 790.75  | 655.10   | 200.56  |
| Assault          | 5400.13 | 4438.82 | 1846.75  | 1363.46 |
| Asterix          | 4310.67 | 3483.17 | 3657.67  | 2651.33 |
| Asteroids        | 2488.10 | 1605.33 | 1615.37  | 2205.70 |
| Atlantis         | 2193605.67 | 2140536.33 | 1515993.33 | 1419104.67 |
| BankHeist        | 1212.23 | 1206.67 | 1124.43  | 1125.17 |
| BattleZone       | 15466.67 | 14766.67 | 14690.00 | 15123.33 |
| BeamRider        | 4594.00 | 2624.19 | 6898.09  | 5073.75 |
| Bowling          | 38.99   | 47.27   | 30.48    | 31.24   |
| Boxing           | 97.23   | 93.70   | 65.33    | 50.07   |
| Breakout         | 458.41  | 281.93  | 67.70    | 40.65   |
| Centipede        | 3315.44 | 3565.18 | 3393.93  | 3353.14 |
| Chopper-Command  | 6308.33 | 4872.67 | 2676.00  | 2286.67 |
| CrazyClimber     | 120247.33 | 105940.00 | 98219.67 | 87522.33 |
| DemonAttack      | 61147.33 | 26740.57 | 57476.65 | 21525.08 |
| DoubleDunk       | -7.89   | -11.22  | -8.61    | -10.04  |
| Enduro           | 459.85  | 698.46  | 518.41   | 365.95  |
| FishingDerby     | 28.99   | 17.72   | -64.27   | -69.64  |
| Freeway          | 21.21   | 21.11   | 18.37    | 20.89   |
| Frostbite        | 316.87  | 280.30  | 280.30   | 291.77  |
| Gopher           | 6207.00 | 1791.00 | 940.87   | 938.27  |
| Gravitar         | 557.17  | 753.50  | 449.00   | 495.17  |
| IceHockey        | -4.12   | -4.83   | -3.61    | -4.61   |
| Jamesbond        | 527.17  | 488.17  | 685.17   | 901.67  |
| Kangaroo         | 3891.67 | 6845.00 | 1850.00  | 1214.67 |
| Krull            | 7715.68 | 8329.08 | 7204.95  | 4881.65 |
| KungFuMaster     | 33728.00 | 29958.67 | 29843.67 | 26808.00 |
| Montezuma-Revenge| 0.00    | 10.67   | 0.67     | 0.00    |
| MsPacman         | 1683.87 | 1981.50 | 1170.70  | 1133.57 |
| NameThisGame     | 6065.63 | 5397.47 | 5672.60  | 5604.10 |
| Pitfall          | 0.00    | -2.32   | -17.26   | -43.60  |
| Pong             | 20.50   | 20.80   | 20.79    | 19.63   |
| PrivateEye       | 79.67   | 36.50   | 99.67    | 99.33   |
| Qbert            | 15396.67 | 14556.83 | 4114.00  | 3781.58 |
| Riverraid        | 8052.23 | 7360.40 | 7722.00  | 6773.67 |
| RoadRunner       | 44679.67 | 36289.33 | 43626.33 | 24061.33 |
| Robotank         | 4.60    | 14.15   | 24.60    | 24.18   |
| Seaquest         | 1807.47 | 1470.60 | 1501.47  | 926.40  |
| SpaceInvaders    | 1216.15 | 944.63  | 814.53   | 634.07  |
| StarGunner       | 48984.00 | 33862.00 | 47738.00 | 33442.67 |
| Tennis           | -8.32   | -13.74  | -19.13   | -18.40  |
| TimePilot        | 3770.33 | 5321.33 | 6278.33  | 5701.00 |
| Tutankham        | 241.21  | 177.58  | 135.80   | 136.21  |
| UpNDown          | 242701.51 | 153160.66 | 11815.87 | 10949.53 |
| Venture          | 36.33   | 0.00    | 4.00     | 0.00    |
| VideoPinball     | 37780.70 | 31577.24 | 21438.64 | 25095.20 |
| WizardOfWor      | 4704.00 | 4886.67 | 3533.67  | 3103.00 |
| Zaxxon           | 9472.00 | 5728.67 | 1179.67  | 4796.67 |

Table 11. Mean final scores (last 100 episodes) of PPO, POP3D, BASELINE and TRPO on Atari games after 40M frames. The results are averaged on three trials.
| game                  | POP3D   | PPO     | BASELINE | TRPO    |
|-----------------------|---------|---------|----------|---------|
| Alien                 | 1147.29 | 1115.94 | 851.13   | 841.08  |
| Amidar                | 299.55  | 413.46  | 295.91   | 169.12  |
| Assault               | 2139.15 | 2168.93 | 1159.50  | 971.78  |
| Asterix               | 2004.43 | 2102.10 | 1884.68  | 1342.83 |
| Asteroids             | 1652.48 | 1470.46 | 1477.71  | 1760.73 |
| Atlantis              | 488134.03 | 596807.27 | 192798.74 | 174394.94 |
| BankHeist             | 662.26  | 643.94  | 859.25   | 831.95  |
| BattleZone            | 11131.44 | 9387.77 | 11674.30 | 12918.39 |
| BeamRider             | 1965.27 | 1460.59 | 3321.25  | 2431.63 |
| Bowling               | 37.97   | 39.41   | 33.90    | 30.99   |
| Boxing                | 83.12   | 78.61   | 27.92    | 23.07   |
| Breakout              | 143.60  | 124.98  | 29.99    | 26.56   |
| Centipede             | 3056.81 | 3344.63 | 3042.48  | 3142.22 |
| Chopper-Command       | 3269.47 | 3106.14 | 1780.38  | 1595.82 |
| CrazyClimber          | 97257.52 | 90169.60 | 69258.31 | 63189.78 |
| DemonAttack           | 7611.27 | 7180.43 | 9814.42  | 6204.68 |
| DoubleDunk            | -13.70  | -15.45  | -15.93   | -14.57  |
| Enduro                | 107.84  | 321.20  | 92.59    | 140.67  |
| FishingDerby          | -21.00  | -27.51  | -81.90   | -81.97  |
| Freeway               | 17.76   | 15.87   | 15.93    | 17.33   |
| Frostbite             | 276.47  | 267.73  | 270.42   | 270.57  |
| Gopher                | 1556.29 | 1196.20 | 900.74   | 875.93  |
| Gravitar              | 413.20  | 509.81  | 342.74   | 317.86  |
| IceHockey             | -4.67   | -5.50   | -4.61    | -5.21   |
| Jamesbond             | 358.54  | 394.45  | 380.91   | 519.01  |
| Kangaroo              | 1614.63 | 2199.74 | 937.98   | 566.85  |
| Krull                 | 6538.16 | 7195.24 | 4760.66  | 3861.87 |
| KungFuMaster          | 23253.96 | 23283.31 | 19637.58 | 18293.12 |
| Montezuma-Revenge     | 3.08    | 8.65    | 13.89    | 14.57   |
| MsPacman              | 1214.09 | 1482.77 | 860.63   | 864.84  |
| NameThisGame          | 5353.14 | 5199.37 | 4562.32  | 4504.67 |
| Pitfall               | -2.41   | -5.81   | -31.27   | -33.93  |
| Pong                  | 13.24   | 12.83   | 7.20     | -2.91   |
| PrivateEye            | 87.37   | 52.76   | 56.70    | 98.79   |
| Qbert                 | 5852.10 | 6744.13 | 1760.92  | 1679.03 |
| Riverraid             | 5260.89 | 5487.17 | 5220.64  | 4549.22 |
| RoadRunner            | 25456.31 | 24688.07 | 20385.91 | 16269.40 |
| Robotank              | 3.08    | 8.65    | 13.89    | 14.57   |
| Seaquest              | 1487.84 | 1120.15 | 1112.51  | 848.47  |
| SpaceInvaders         | 693.26  | 632.17  | 552.50   | 483.48  |
| StarGunner            | 14734.11 | 13643.80 | 16288.35 | 13341.23 |
| Tennis                | -19.86  | -21.80  | -21.84   | -21.04  |
| TimePilot             | 3396.61 | 4410.87 | 4718.46  | 4544.68 |
| Tutankham             | 179.96  | 152.72  | 103.95   | 109.18  |
| UpNDown               | 38728.48 | 43208.99 | 5430.22  | 7085.02 |
| Venture               | 15.89   | 14.66   | 0.57     | 0.03    |
| VideoPinball          | 27346.44 | 27549.55 | 23998.09 | 23705.39 |
| WizardOfWor           | 2340.60 | 2743.40 | 2409.94  | 2045.17 |
| Zaxxon                | 3739.56 | 1813.90 | 256.78   | 1521.28 |

Table 12. All episodes mean scores of PPO, POP3D, BASELINE and TRPO on Atari games after 40M frames. The results are averaged by three trials.