Empirical Analysis of Policy Gradient Algorithms where Starting States are Sampled accordingly to Most Frequently Visited States

Samy Aittahar * **Raphaël Fonteneau** * **Damien Ernst** *

* *University of Liège, Belgium*
{saittahar, raphael.fonteneau, dernst}@uliege.be

Abstract
In this paper, we propose an extension to the policy gradient algorithms by allowing starting states to be sampled from a probability distribution that may differ from the one used to specify the reinforcement learning task. In particular, we suggest that, between policy updates, starting states should be sampled from a probability density function which approximates the state visitation frequency of the current policy. Results generated from various environments clearly demonstrate a performance improvement in terms of mean cumulative rewards and substantial update stability compared to vanilla policy gradient algorithms where the starting state distributions are either as specified by the environment or uniform distributions over the state space. A sensitivity analysis over a subset of the hyper-parameters of our algorithm also suggests that they should be adapted after each policy update to maximise the improvements of the policies.

Keywords: Reinforcement learning control

1. INTRODUCTION

Over the last few years, a wide variety of on-policy search algorithms for reinforcement learning (RL) - for which the framework is explained in details in Section 2 - have been designed to find policies that maximise the expected long-term return in challenging continuous environments where starting states are sampled from a predefined distribution (Hessel et al. [2017], Mnih et al. [2015], Schulman et al. [2017]). These algorithms proceed according to the following principle: from interactions with its surrounding environment, an agent updates, and hopefully improves, its decision strategy using instantaneous collected rewards. The agent may repeat such a procedure ad infinitum, eventually discovering decision strategies that maximise collected rewards.

Policy search algorithms mainly differ through their update rules. More specifically, and whenever the policy is differentiable and parametrized - which is a common assumption -, it is possible to design gradient-based update rules using appropriate differentiable loss functions to maximise the policy expected return (Gu et al. [2016], Lillicrap et al. [2015], Schulman et al. [2015, 2017], Wang et al. [2016]). In case of a non-differentiable policy, gradient-free update rules may be considered, such as genetic algorithms (Such et al. [2017]), simulated annealing (Atiya et al. [2004]) or cross-entropy methods (Mannor et al. [2003], Busoniu et al. [2010]). In this paper, we focus on the assumption that the policies are differentiable and parametrized.

In policy gradient algorithms, (see Section 3 for a more detailed description) the agents update their policies with their gradient estimates that are computed using expected return samples for each state-action pair that they have encountered while playing their policies. MC-REINFORCE is a classical example of such an approach (Williams [1992]). These algorithms are simple to implement, but exhibit high gradient variance on policy updates, especially when the amount of interactions is limited (Weaver and Tao [2013]). State-of-the-art policy gradient algorithms, often relying on additional function approximators to reduce the gradient variance, have greatly helped to solve challenging tasks in complex environments (Hessel et al. [2017], Mnih et al. [2015], Schulman et al. [2015], Schulman et al. [2017]), even though learning structures may be tricky to tune (Henderson et al. [2017]).

Recent developments in RL have proposed to dynamically modify starting state distributions to improve the learning process. Approaches proposed by Salimans and Chen [2018] and Popov et al. [2017] uniformly sample the starting states from a set of states gathered from expert demonstrations, whereas Florensa et al. [2017] initially samples starting states close to goal states, and then progressively samples states further around. Although showing performance as well as variance reduction improvements, these works rely on the availability of either goal states (Florensa et al. [2017]) or previously encountered states (Florensa et al. [2017], Popov et al. [2017], Salimans and Chen [2018]). As an alternative, following the observation that the gradient variance is aggravated by a scarcity of state-action pairs which is amplified over the interaction course of the agent, especially when the number of episodes is low, and following the intuition that this amplification might be mitigated by favouring the exploration of states that are often visited by the agent, we propose to maintain an estimation of the state visitation frequency of the policy from which we sample starting states before each episode. The derived algorithm, named MCP0-REINFORCE, works as follows. At the very first episode, the first state is sampled from the true starting state distribution of the environment, and the episode is generated by the agent with a prior policy. At the end of the episode, the collected set of states observed so far is used to estimate a parametric probability density function (PDF) over the state space. From this point, an episode rolled by the current policy always starts by a state that has been sampled from a parametric PDF for which parameters maximise the likelihood of states already encountered by the agent with its current policy in previous episodes. When the agent has generated the last episode, the parameters of the PDF are updated to maximise the likelihood of all states encountered by the agent so far and its policy is updated exactly like in the MC-REINFORCE algorithm. To continue the policy updates, the MCP0-REINFORCE is executed again exactly as described above, but the prior policy is replaced by the updated policy and the first state of the first episode is sampled from the last parametric PDF which approximates the state visitation frequency of the previous policy, and so on. The MCP0-REINFORCE algorithm is explained in Section 4.

In Section 5, we empirically show that our method allows to get better performances, with more stable policy updates, than the vanilla policy gradient algorithm for which the starting state distribution is either the one specified by the environment or an uniform distribution over the state space. Furthermore, we provide a sensitivity analysis on MCP0-REINFORCE over a subset of its hyper-parameters which suggests that the algorithm is rather sensitive to them and they should be optimized through the policy search course. We conclude, in Section 6, by a discussion on possible future research directions to overcome the limitations of our approach.

2. RL BACKGROUND

We consider continuous control tasks which may be represented by a discrete-time Markov Decision Process (MDP). We consider MDPs with a continuous state space $\mathcal{S}$, a continuous action space $\mathcal{U}$, a probability distribution over the starting states $s_0 \sim p_0(\cdot)$, a probability density function (PDF) for generating subsequent states s_{t+1} from any pair $(s_t, u_t) \in \mathcal{S} \times \mathcal{U}$, i.e. $s_{t+1} \sim P(\cdot|s_t, u_t)$ and an instantaneous real-valued reward function ρ associating, for any transition (s_t, u_t, s_{t+1}), a real value $\rho(s_t, u_t, s_{t+1}) \in \mathbb{R}$. All rewards are assumed to be bounded, i.e. that $\exists \rho^* \in \mathbb{R}^+ | \forall (s, s', u) \in \mathcal{S}^2 \times \mathcal{U}, |\rho(s, u, s')| \leqslant \rho^*$. State and action spaces are respectively assumed to be complete, separable metric Polish spaces equipped with σ-algebras $(d_\mathcal{S}, \sigma_\mathcal{S})$ and $(d_\mathcal{A}, \sigma_\mathcal{A})$. We also assume that an agent may interact with the MDP but does not explicitly know its components.

Let Π denote the set of state-dependent stochastic policies. Given a policy $\pi \in \Pi$ and a state $s \in \mathcal{S}$, a decision $u \in \mathcal{U}$ can be drawn according to π given s, i.e. $u \sim \pi(\cdot|s)$, where $\pi(\cdot|s)$ is assumed to be a PDF. An agent starting from a given initial state $s_0 \in \mathcal{S}$, taking an initial action $u_0 \in \mathcal{U}$ and following π afterwards will observe an infinite stochastic episode of states and actions $E^\pi_{s_0,u_0} = (s_0, u_0, s_1 \sim P(\cdot|s_0, u_0), u_1 \sim \pi(\cdot|s_1), \ldots)$ from which a truncated discounted return from $t_0 \in \mathbb{N}$ to $t_0 + k - 1$ can be computed:

$$R^{E^\pi_{s_{t_0},u_{t_0}}}_{t_0 \ldots t_0+k-1} = \sum_{t=t_0}^{t_0+k-1} \gamma^{t-t_0} \rho(s_t, u_t, s_{t+1}), \tag{1}$$

where $\gamma \in [0, 1[$ is the discount factor, which defines the sight of the expected observable long-term return. The expected return in following a stochastic policy π starting from state s, also called the value function, is defined as follows:

$$V^\pi(s) = \underset{\substack{u_t \sim \pi(\cdot|s_t), \forall t \in \mathbb{N} \\ s_{t+1} \sim P(\cdot|s_t,u_t), \forall t \in \mathbb{N}}}{\mathbb{E}} \left\{ \lim_{k \to \infty} R^{E^\pi_{s_0,u_0}}_{0 \ldots k-1} \middle| s_0 = s \right\}. \tag{2}$$

Where considering stochastic policies in model-free reinforcement learning problems, it is common practice to compute the expected return of following a policy after having taken action u in state s through a Q-function:

$$Q^\pi(s, u) = \underset{\substack{u_t \sim \pi(\cdot|s_t), \forall t \in \mathbb{N} \setminus \{0\} \\ s_{t+1} \sim P(\cdot|s_t,u_t), \forall t \in \mathbb{N}}}{\mathbb{E}} \left\{ \lim_{k \to \infty} R^{E^\pi_{s_0,u_0}}_{0 \ldots k-1} \middle| \begin{matrix} s_0 = s \\ u_0 = u \end{matrix} \right\}. \tag{3}$$

Using the Q-function defined in Equation 3, the overall expected return of the policy is computed as follows:

$$J(\pi) = \underset{\substack{s \sim p_0(\cdot) \\ u \sim \pi(\cdot|s)}}{\mathbb{E}} \{Q^\pi(s, u)\}. \tag{4}$$

The objective of the agent is to find a policy $\pi^* \in \arg\max_{\pi \in \Pi} J(\pi)$ that maximises the overall expected discounted return defined in Equation 4. Since the environment is continuous and the agent does not know explicitly the components of the MDP, the optimization problem $\pi^* \in \arg\max_\pi J(\pi)$ is intractable. Among the many approaches to compute an approximation of π^*, we focus on policy gradient algorithms which can be assimilated to simulation-based gradient-ascent procedures on differentiable policies.

3. POLICY GRADIENT ALGORITHMS

Policy gradient algorithms, in their core principle, iteratively update differentiable policies with gradient estimates of Equation 4 computed through their sequential interactions with the environment. As long as the number of sequential interactions with the environment grows, these estimates are getting closer to the true gradients of the policies.

The Policy Gradient Theorem, stated as follows and provided by Sutton et al. [2000], shows how to analytically compute the gradient of Equation 4:

$$\nabla_\theta J(\pi_\theta) = \int_\mathcal{S} \int_\mathcal{U} p^{\pi_\theta}(s) Q^{\pi_\theta}(s, u) \nabla_\theta \pi_\theta(u|s) du ds, \tag{5}$$

where

$$p^{\pi_\theta}(s) = \int_\mathcal{S} p_0(s') \lim_{T \to \infty} \sum_{k=0}^{T} \gamma^k Pr(s' \to s, k, \pi_\theta) ds', \tag{6}$$

π_θ is a differentiable policy parametrized by θ, $P_r(s \to s', k, \pi_\theta)$ computes the (density) probability of reaching s' from s in $k \geqslant 0$ steps by following policy π_θ and p^{π_θ} is the unnormalized discounted future state visitation distribution of state s.

Since Q^{π_θ} and p^{π_θ} are unknown by advance, Equation 5 is intractable. A common approach is to approximate the gradient through a sampled-based estimator. Let $l \in \{1, \ldots, L\}$ be an index for identifying any truncated episode $E^{\pi_\theta,l}_{s_t,u_t}$ obtained when playing policy π_θ from the state-action pair (s_t, u_t). The gradient $\nabla_\theta J(\pi_\theta)$ defined by Equation 5 is estimated as follows:

$$\widehat{\nabla}_\theta J(\pi_\theta) = \frac{1}{Z} \sum_{l=1}^{L} \sum_{k=0}^{T_l-1} \nabla_\theta \; log \; \pi_\theta(u_k|s_k) R_{k\ldots T_l-1}^{E^{\pi_\theta,l}_{s_k,u_k}}, \qquad (7)$$

where T_l is the length of the truncated episode $E^{\pi_\theta,l}_{s_t,u_t}$ and $Z = \sum_{l=1}^{L} T_l$.

As shown by Williams [1992], Equation 7 is an unbiased estimator of Equation 5. It is called the REINFORCE estimator. The derived algorithm, namely MC-REINFORCE, is also detailed by Williams [1992]. The main caveat of MC-REINFORCE is the high variance over policy updates (Sutton et al. [2000]), especially when the sampling capacity is limited. This is mainly due to the variation of the gradient estimates, which is aggravated by (i) the dependency to the starting state distribution, (ii) the stochastic transition function of the environment and (iii) the sampling policy.

The next section describes how we attempt to mitigate the first of the three above-mentioned issues by explicitly focusing exploration on state regions that are more frequently visited by the policy. To that extent, we assume that the agent is able to modify the starting state distribution and exploits this ability to favor visitation of state regions that it frequently encounters during the exploration phase.

4. MCP0-REINFORCE

The gradient variance of MC-REINFORCE is particularly sensitive to a low number of available episodes. Indeed, a low number of episodes implicitly generates a scarcity of state-action pairs along the trajectory generated by the policy π_θ, which dramatically decreases the quality of the gradient estimate. At the first steps of the trajectory, this scarcity is mainly due to the entropy of the starting state distribution, which is often defined as a uniform distribution over a subset of the state space. Over the course of the trajectory, this scarcity is aggravated by both the entropy of the policy and of the subsequent states probability density function. In such situations, the state distribution of the trajectories is often drastically different of the state visitation frequency of the policy, which has a substantial impact on the gradient estimate error.

Following the above-mentioned discussion, we propose to use the discounted future state visitation PDF $p^{\pi_\theta}(\cdot)$ as starting state distribution. Unfortunately, $p^{\pi_\theta}(\cdot)$ cannot be computed analytically. We propose a learning-based approach by estimating a parametric density function over the state space aiming at representing the discounted future state visitation PDF of the policy π_θ.

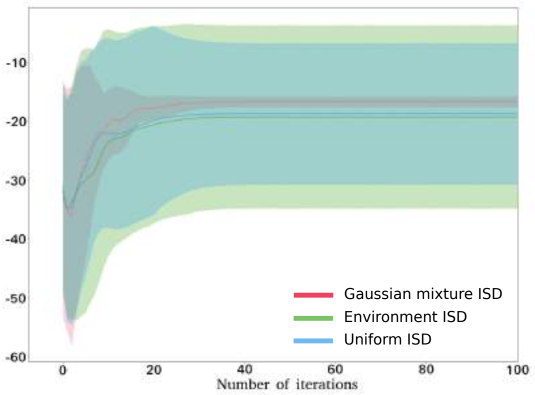


Figure 1. Mass Spring Damper - Mean undiscounted cumulative reward over 30 simulations on cases described in Section 5. *Environment ISD*, *Uniform ISD* and *Gaussian mixture ISD* refer to the first, second and third cases, respectively.

To implement this approach into the MC-REINFORCE algorithm without increasing the sample complexity, we propose the following new policy gradient algorithm, namely MCP0-REINFORCE. Before running the very first episode with a prior policy π_θ, the starting state is sampled by the original probability distribution $p_0(\cdot)$. Once the episode is finished, the parameters of a predefined PDF are computed to maximise the likelihood of states encountered during the episode. Before running the second episode with the same policy, the starting state is sampled from this parametric PDF. Once the episode is finished, the parameters of the PDF are computed to maximise the likelihood of states of the *two* episodes. Then, the starting state of the third episode is sampled from the newly computed PDF, and the process continues like this until the end of the last episode, for which the starting state has been sampled from the PDF parametrized to maximise the likelihood of the states encountered during all the previous episodes. The policy is then updated exactly like in the MC-REINFORCE algorithm, and the MCP0-REINFORCE algorithm is launched again by setting (i) the new policy as the prior policy and (ii) the starting state distribution as the PDF for which the parameters maximise the likelihood of all the states encountered by the previous policy.

In the next section, empirical benchmarks for MC-REINFORCE and MCP0-REINFORCE are discussed and reported through a variety of environments. We also propose a sensitivity analysis over the initial learning rate, the number of episodes and the hyper-parameters of the PDF structure used to sample starting states in MCP0-REINFORCE.

5. RESULTS

All the environments share the following characteristics in their experimental protocols, which are detailed in their respective subsections. The policy is represented by two differentiable function approximators, which are parametrized linear combinations of 500 random Fourier feature mappers (RFFM) extracted from state vectors

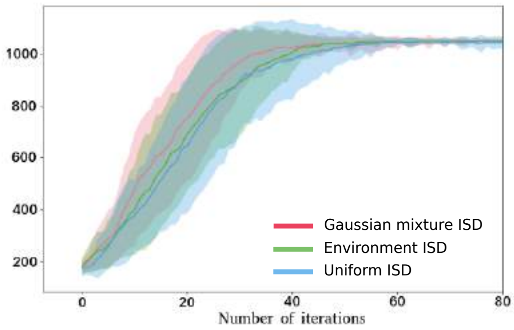


Figure 2. Catcher Game - Mean undiscounted cumulative reward (a) and variance on undiscounted cumulative rewards (b) over 30 simulations on cases described in Section 5. *Environment ISD*, *Uniform ISD* and *Gaussian mixture ISD* refer to the first, second and third cases, respectively.

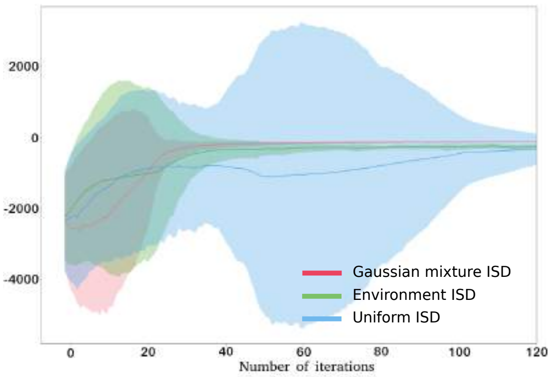


Figure 3. Ship Steering - Mean undiscounted cumulative reward over 30 simulations on cases described in Section 5. *Environment ISD*, *Uniform ISD* and *Gaussian mixture ISD* refer to the first, second and third cases, respectively.

(see a detailed discussion provided by Rahimi and Recht [2007]). Their outputs form the parameters of a multivariate normal distribution, where the first output is the mean vector and the second one is the variance diagonal vector - post-processed with a shifted exponential function. A new learning rate is computed at each policy update - also called an iteration - by the ADAM optimizer (Kingma and Ba [2014]) from an initial learning rate which is provided in the specific settings. The collected expected return estimates are centered around zero instead of their true means, as they can be always positive or negative, and still leaves the gradient estimator unbiased. Discount factor and time horizon values, which are set differently for each environment, are motivated by a trade-off between the sight defined by the discount factor and the potential gradient variance which is aggravated by the time horizon.

To assess the improvements raised by our approach, we have designed three cases. The first and the second cases run the MC-REINFORCE algorithm while starting states are sampled from the distribution specified by the environment and an uniform distribution over the state space, respectively. The third case runs the MCP0-REINFORCE algorithm which replaces the starting state distribution by a finite mixture of multivariate Gaussian distributions - also called Gaussian mixture models, as McLachlan and Peel [2000] have shown that they are universal function approximators - which is built at the end of each episode by providing the episodic history to an implementation of the Expectation-Maximisation algorithm from the *scikit-learn* [1] programming library. Specific settings for this estimation are outlined for each environment. Whenever the starting state is sampled beyond the state space, the variables are clipped on their respective bounds in a way that is specified by the environment state space.

For each environment, at each iteration, the cases are compared by their results in terms of truncated mean cumulative rewards through simulations which are run independently on 30 different random number generators (RNG). For each case, each simulation and after each policy update, the mean truncated return is computed by running the policy through 30 episodes for which starting states are sampled from the distribution specified by the environment. Shaded areas in the plots are standard deviations over the mean truncated cumulative rewards computed over the different RNGs. For all cases, these results are gathered by simulating the different policies from starting states sampled by the true probability distribution p_0.

The following subsections describe in details the results for each environment. In order to ease the reproducibility of the experiments - following the recommandations provided by Henderson et al. [2017] - the full implementation is available online [2]. We also provide, for the second and third subsections, a sensitivity analysis for the third case over (i) the number of episodes rolled by the policy before update, (ii) the initial learning rate and (iii) the number of components in the mixture of Gaussian distributions.

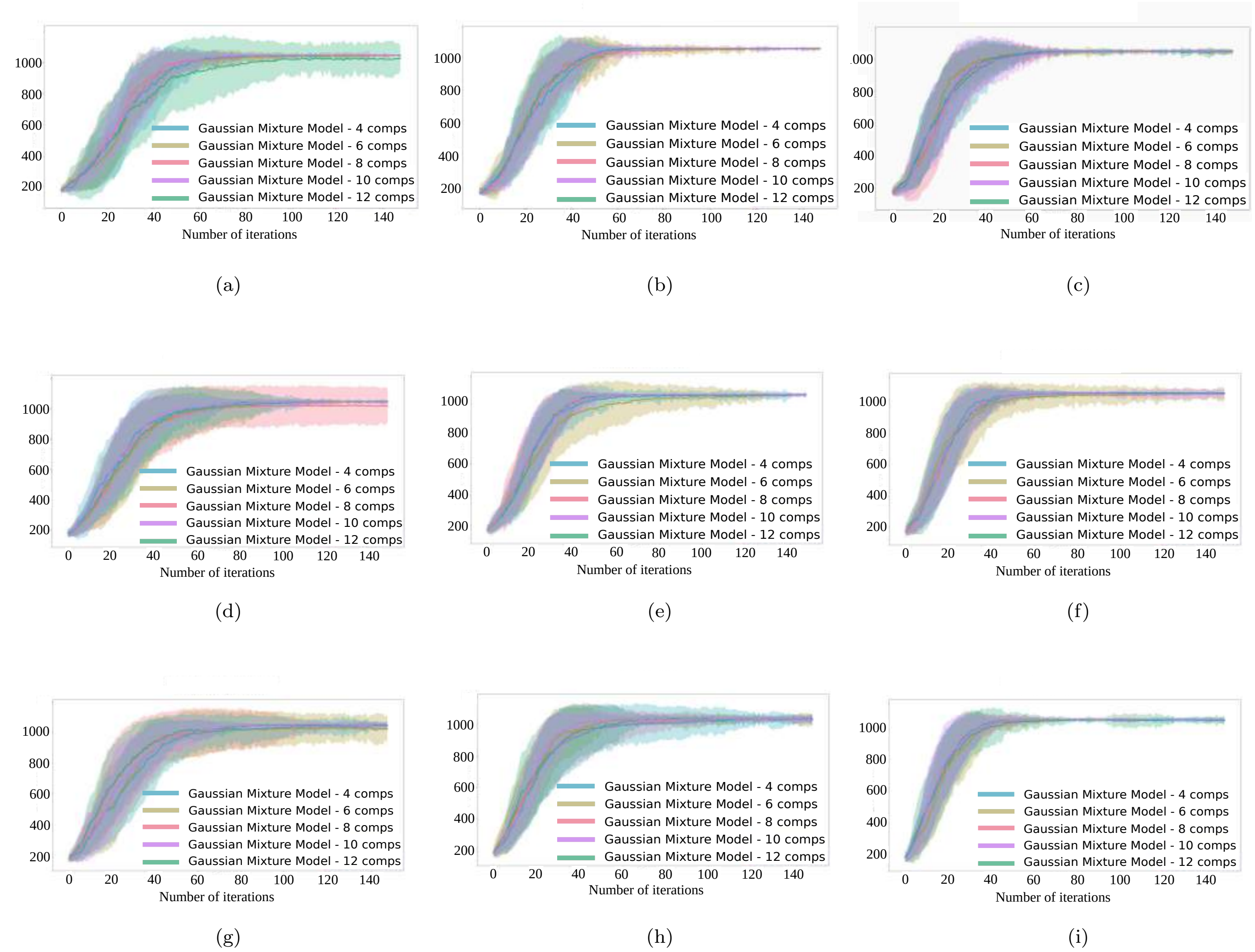
5.1 Mass Spring Damper

The Mass Spring Damper system is a simple mechanical system which is popular in reinforcement learning benchmarks (Bou Ammar and Taylor [2012], Pirotta and Bascetta [2015]). The experimental protocol is the following. The RFFM inner weights and bias are both randomly and independently sampled by following the $\mathcal{N}(0, \frac{1}{10})$ distribution. Weights and bias of the linear combination are initialized with a distribution following $\mathcal{N}(0, \frac{1}{5})$. The discount factor is set to 0.9 and the horizon time T is set to 40 - these values are also considered by Pirotta and Bascetta [2015]. The initial learning rate α is 0.05, which decreases linearly along the number of iterations. At each iteration, 4 episodes are generated by sampling actions from the policy and used to estimate its gradient. The parameters of the starting state distribution, if relevant, are updated at the end of each episode, using the episodes previously generated by the current policy. The number of components of the mixture of Gaussian distributions is set to 5.

Figure 1 shows the results obtained by running the experimental protocol aforementioned. We can notice that the mean cumulative reward of the policies updated during the

[1] See `https://scikit-learn.org`

[2] See https://github.com/epochstamp/mcp0



(a) (b) (c)

(d) (e) (f)

(g) (h) (i)

Figure 4. Catcher Game - Mean cumulative reward with learning rates 0.05 (a,b,c), 0.07 (d,e,f) and 0.09 (g,h,i) over 30 simulations. From left to right column, the number of training episodes are 6, 8 and 10, respectively.

three scenarios steadily seems to converge to slightly different values. This type of behaviour was expected because of the decreasing learning rate and thus to the vanishing policy updates. We note also that the standard deviation is small on the third case, while conversely it is much higher on the two first cases. The positive effect of the bias in the policy updates induced by the third case is clearly visible after a few policy updates in the context of this environment.

5.2 *Catcher Game*

The Catcher Game has been adapted from an external programming library by replacing discrete actions with continuous actions [3]. The experimental protocol is the following. The RFFM inner weights and bias are both randomly and independently sampled by following the $\mathcal{N}(0, \frac{1}{10})$ distribution. Weights and bias of the linear combination are initialized with a distribution following $\mathcal{N}(0, \frac{1}{5})$. The horizon time T is set to 1000 and the discount factor is set to 0.95. The dynamics of the environment are stochastic, which potentially introduces more variance in the gradient estimate. The learning rate α is 0.09. At each iteration, 10 episodes are generated by sampling actions from the policy and used to estimate its gradient. The number of components of the mixture of Gaussian distributions is set to 9. The parameters of the starting state distribution, if relevant, are updated at the end of each episode, using the whole history of the current iteration. The parameters which are modified for the sensitivity analysis are the number of episodes which are taken from $\{6, 8, 10\}$, the initial learning rates from $\{0.05, 0.07, 0.09\}$ and the number of components of the Gaussian mixture model from $\{4, 6, 8, 10, 12\}$.

Figure 2 shows the results raised by the experimental protocol aforementioned. We can first notice that the mean cumulative return of all the policies across cases are converging to similar values. Furthermore, it can be observed that during the learning process, the mean cumulative reward is slightly higher in the third case than the two others cases, and that the standard deviation over the mean cumulative rewards is slightly lower.

Figures 4 shows the results in terms of mean cumulative reward obtained by running the experimental protocol which

[3] See `https://pygame-learning-environment.readthedocs.io`

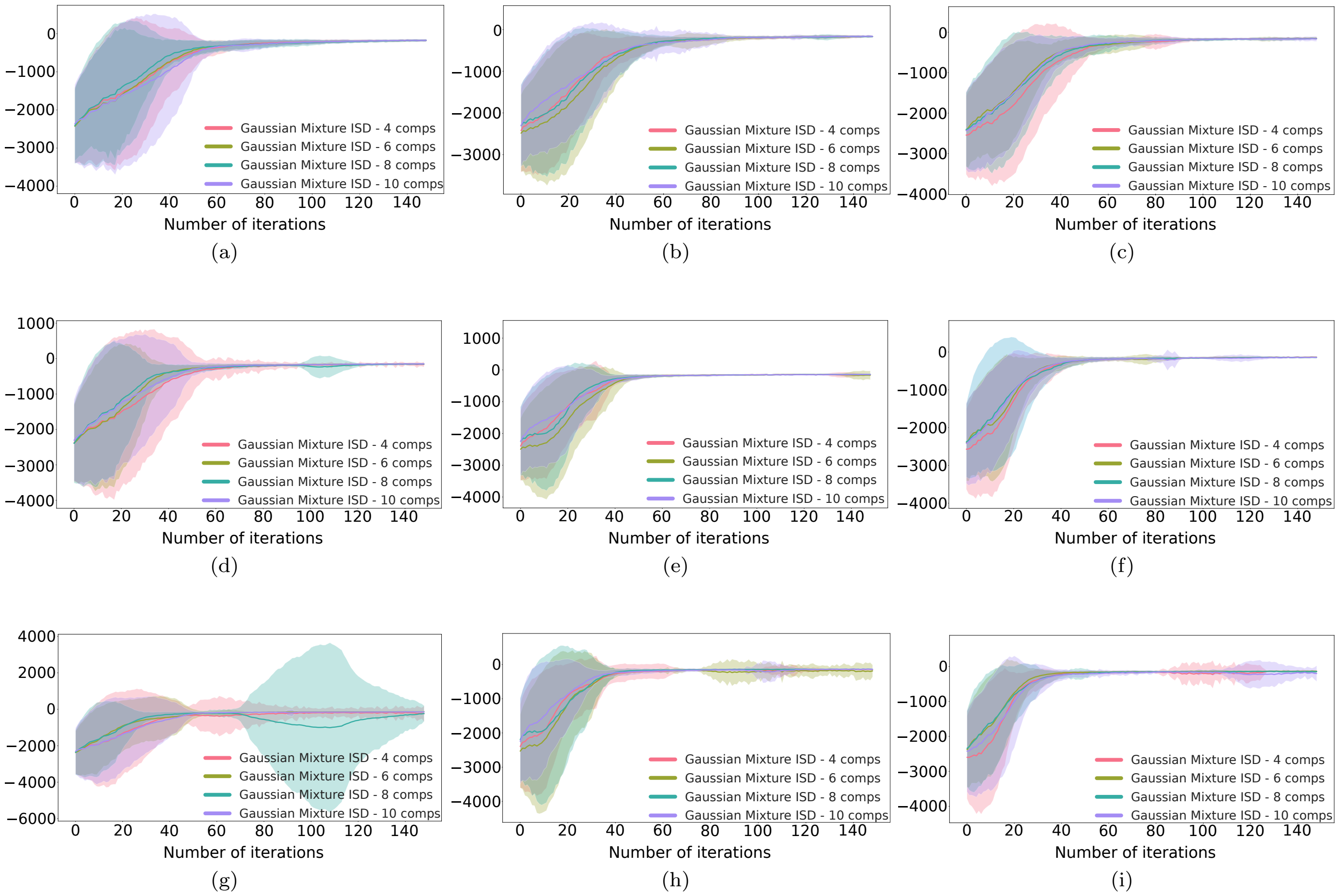


Figure 5. Ship Steering - Mean cumulative reward with learning rates 0.005 (a,b,c), 0.007 (d,e,f) and 0.009 (g,h,i) over 30 simulations. From left to right column, the number of training episodes are 20, 40 and 50, respectively.

each combination of hyper-parameters set. These results show that the performances of the MCP0-REINFORCE algorithm is rather sensitive to both the number of components of the finite Gaussian mixture and the initial learning rate.

For instance, when 6 episodes are rolled by the policy, and with an initial learning rate of 0.05, it seems that running the third case with a mixture of 8 Gaussian distributions leads to slightly higher mean cumulative rewards than the others mixtures of Gaussian distributions, though the mixture of 10 Gaussian distributions yields similar performances. On the other hand, by increasing the learning rate up to 0.07, it can be observed that running the third case again with a mixture of 8 Gaussian distributions to slightly lower mean cumulative rewards than with the mixture of 10 Gaussian distributions and to a sensitively higher standard deviation. Following these observations, if a higher learning rate is chosen, we should expect worse performances for the third case using a mixture of 8 Gaussian distributions. However, our results show instead that the performances are close to the best configuration - which uses a mixture of 12 Gaussian distributions - while the others ones show lower performances at the 60 first policy updates.

This phenomenon is also observed when the policy is rolled through a higher number of episodes, even though the differences are less visibles, which is expected since the variance over the policy updates mechanically reduces as long as the number of episodes grows. However, we can observe that, with the highest number of episodes (30) and the highest learning rate (0.09), running the third case with a mixture of 6 Gaussian distributions leads to slightly higher mean cumulative rewards than the others mixtures during the first policy updates, and to slightly lower mean cumulative rewards with sensitively higher standard deviations over mean cumulative rewards compared to the others mixtures.

5.3 Ship Steering

The Ship Steering system is a non-linear system considered as a reinforcement learning benchmark by Pirotta and Bascetta [2015]. The experimental protocol is the following. The RFFM inner weights and bias are both randomly and independently sampled by following the $\mathcal{N}(0, \frac{1}{10})$ distribution. Weights and bias of the linear combination are initialized with a distribution following $\mathcal{N}(0, \frac{1}{5})$. The horizon time T is set to 30 and the discount factor is set to 0.6. The learning rate α is 0.009. At each iteration, 10 episodes are generated by sampling actions from the policy and used to estimate its gradient. The number of components of the Gaussian mixture is set to 9. The parameters of the starting state distribution, if relevant, are updated every two episodes, using the whole history of the current iteration. The parameters which are modified for the sensitivity analysis are the number of episodes which are taken from $\{20, 40, 50\}$, the initial learning rates from $\{0.005, 0.007, 0.009\}$ and the number of components of the Gaussian mixture model from $\{4, 6, 8, 10\}$.

Figure 3 shows the results obtained by running the experimental protocol aforementioned. It is interesting to notice how the second case - the uniform distribution - exhibits a huge error in the middle of policy updates for quite some time before eventually decreasing. On the other hand, the third case eventually converges to a slightly higher mean cumulative reward than the first one. The third case, despite having a lower mean cumulative reward over the first policy updates, eventually gets higher mean cumulative rewards than the two others cases. Finally, the first and the third cases eventually converge to similar mean cumulative rewards and standard deviations over them, while the performances of the second case are dramatically worse than for the others cases.

Figure 5 show the results in terms of mean cumulative reward obtained by running the experimental protocol which each combination of parameters set. These results show that the performances of the MCP0-REINFORCE algorithm are highly sensitive to the number of components of the finite Gaussian mixtures. For example, when the number of episodes is 20, and the learning rate is 0.005, running the third case with a mixture of 8 Gaussian distributions leads to higher rewards and lower standard deviations over the mean cumulative rewards than the others mixtures. But when the learning rate is higher (0.009), running the third case with a mixture of 8 Gaussian distribution leads this time to a substantially higher standard deviation over the mean cumulative rewards after the $80th$ policy update, whereas mean cumulative rewards were higher than the others configurations during the first policy updates, though the performances are similar to the third case with a mixture of 6 Gaussian distributions.

6. DISCUSSION AND FUTURE WORK

In this paper, we have proposed to modify the policy gradient algorithm under the assumption that the agent may arbitrarily modify the starting state distribution. More specifically, we have proposed to replace the original starting state distribution by an approximation of the discounted future state visitation PDF of the policy. This new approach has been tested on fully continuous environments and compared with cases where starting state distributions were (i) as specified by the environment and (ii) uniform distributions over the state space.

According to the results, it seems that the bias induced by this particular modification of the starting state distribution might have a positive effect on the policy updates, i.e. by increasing the mean cumulative reward and decreasing the gradient variance over the policy updates. Theoretical results related to the bias and the variance of the gradient estimator of the policy where starting states are sampled from the true discounted future state visitation PDF is part of the future research work.

We have also provided extended results of our approach to try to understand how to set the hyper-parameters related to both the parametric steady-state PDF structure and the policy gradient algorithm. These results suggest that the number of components of the Gaussian mixture is not only dependent of the environment, but also mainly dependent on the learning rate and the number of episodes that the agent can generate with its sample policy. While the agent can not play more episodes than allowed, the number of components and the learning rate could still be adapted after each policy update to maximise the improvements of the policy of the agent. Unfortunately, and especially when the task is getting harder, it is often difficult to provide a simple and analytical strategy to that end. Instead, we suggest that a simulation-based optimization problem should be formalised, so that its resolution may lead to a dynamic strategy for choosing hyper-parameters structure and values leading to the maximisation of improvements and stability of the policy updates. Among the possible techniques that may be used to formalize such a strategy, multi-armed bandits approaches (R.Munos [2014]), where arms would be associated to instanced hyper-parameters structures, could offer interesting results, in particular for guiding the hyper-parameters space exploration in a context where the budget for sampling episodes is limited (Trovo et al. [2016]).

Nonetheless, the current approach clearly shows the following limitation in regards of the state-of-the-art reinforcement learning challenges. The implicit assumption behind the MCP0-REINFORCE algorithm is that the agent *knows* the state space as specified by the environment. Unfortunately, in most challenging environments, the agent does not have access to the state space, but instead to surrogate representations (e.g., as images and temporal sequences). In this configuration, the agent is constrained to start interacting with the environment from states that have been sampled by the starting state distribution specified by the environment. An alternative approach could be to introduce a third-party policy which could be trained to quickly reach the most probable starting states that would be drawn by the discounted future state visitation PDF before deploying the actual policy. In this approach, as the required expressive power in practice might be higher as the capacity of the mixture of Gaussian distributions to have a decent estimate of the state visitation frequency of the policy from the corresponding surrogate representations, it could be interesting to consider deep neural networks for this task, for which recent developments to transform them into PDFs are good candidates (Huang et al. [2018], Wehenkel and Louppe [2019]).

Finally, as a last future works proposition, let us underline the fact that, although this paper has focused on policy gradient algorithms, our method could perfectly be applied to gradient-free reinforcement learning algorithms.

REFERENCES

Atiya, A., Parlos, A., and Ingber, L. (2004). A reinforcement learning method based on adaptive simulated annealing. volume 1, 121 – 124 Vol. 1. doi:10.1109/MWSCAS.2003.1562233.

Bou Ammar, H. and Taylor, M.E. (2012). Reinforcement learning transfer via common subspaces. In P. Vrancx,

M. Knudson, and M. Grześ (eds.), *Adaptive and Learning Agents*, 21–36. Springer Berlin Heidelberg, Berlin, Heidelberg.

Busoniu, L., Ernst, D., De Schutter, B., and Babuska, R. (2010). Cross-entropy optimization of control policies with adaptive basis functions. *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, 41(1), 196–209.

Florensa, C., Held, D., Wulfmeier, M., and Abbeel, P. (2017). Reverse curriculum generation for reinforcement learning. *CoRR*, abs/1707.05300. URL `http://arxiv.org/abs/1707.05300`.

Gu, S., Lillicrap, T., Sutskever, I., and Levine, S. (2016). Continuous deep Q-learning with model-based acceleration. *ArXiv e-prints*.

Henderson, P., Islam, R., Bachman, P., Pineau, J., Precup, D., and Meger, D. (2017). Deep reinforcement learning that matters.

Hessel, M., Modayil, J., Hasselt, H.V., Schaul, T., Ostrovski, G., Dabney, W., Horgan, D., Piot, B., Azar, M.G., and Silver, D. (2017). Rainbow: combining improvements in deep reinforcement learning. *CoRR*, abs/1710.02298. URL `http://arxiv.org/abs/1710.02298`.

Huang, C., Krueger, D., Lacoste, A., and Courville, A.C. (2018). Neural autoregressive flows. *CoRR*, abs/1804.00779. URL `http://arxiv.org/abs/1804.00779`.

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

Lillicrap, T.P., Hunt, J.J., Pritzel, A., Heess, N., Erez, T., Tassa, Y., Silver, D., and Wierstra, D. (2015). Continuous control with deep reinforcement learning. *ArXiv e-prints*.

Mannor, S., Rubinstein, R.Y., and Gat, Y. (2003). The cross entropy method for fast policy search. In *Proceedings of the 20th International Conference on Machine Learning (ICML-03)*, 512–519.

McLachlan, G.J. and Peel, D. (2000). *Finite mixture models*. Wiley Series in Probability and Statistics, New York.

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

Pirotta, M. Restelli, M. and Bascetta, L. (2015). Policy gradient in Lipschitz Markov decision processes. *Machine Learning*, 100(2), 255–283. doi:10.1007/s10994-015-5484-1. URL `https://doi.org/10.1007/s10994-015-5484-1`.

Popov, I., Heess, N., Lillicrap, T.P., Hafner, R., Barth-Maron, G., Vecerik, M., Lampe, T., Tassa, Y., Erez, T., and Riedmiller, M.A. (2017). Data-efficient deep reinforcement learning for dexterous manipulation. *CoRR*, abs/1704.03073. URL `http://arxiv.org/abs/1704.03073`.

Rahimi, A. and Recht, B. (2007). Random features for large-scale kernel machines. In *Proceedings of the 20th International Conference on Neural Information Processing Systems*, NIPS'07, 1177–1184. Curran Associates Inc., USA. URL `http://dl.acm.org/citation.cfm?id=2981562.2981710`.

R.Munos (2014). From bandits to Monte-Carlo Tree Search: the optimistic principle applied to optimization and planning. *Foundations and Trends® in Machine Learning*, 7(1), 1–129.

Salimans, T. and Chen, R. (2018). Learning Montezuma's Revenge from a single demonstration. *CoRR*, abs/1812.03381. URL `http://arxiv.org/abs/1812.03381`.

Schulman, J., Levine, S., Moritz, P., Jordan, M.I., and Abbeel, P. (2015). Trust region policy optimization. *ArXiv e-prints*.

Schulman, J., Moritz, P., Levine, S., Jordan, M.I., and Abbeel, P. (2015). High-dimensional continuous control using generalized advantage estimation. *CoRR*, abs/1506.02438. URL `http://arxiv.org/abs/1506.02438`.

Schulman, J., Wolski, F., Dhariwal, P., Radford, A., and Klimov, O. (2017). Proximal policy optimization algorithms. *ArXiv e-prints*.

Such, F.P., Madhavan, V., Conti, E., Lehman, J., Stanley, K.O., and Clune, J. (2017). Deep neuroevolution: genetic algorithms are a competitive alternative for training deep neural networks for reinforcement learning. *CoRR*, abs/1712.06567. URL `http://arxiv.org/abs/1712.06567`.

Sutton, R.S., McAllester, D.A., Singh, S.P., and Mansour, Y. (2000). Policy gradient methods for reinforcement learning with function approximation. In *Advances in neural information processing systems*, 1057–1063.

Trovo, F., Paladino, S., Restelli, M., and Gatti, N. (2016). Budgeted multi–armed bandit in continuous action space. In *Proceedings of the Twenty-second European Conference on Artificial Intelligence*, 560–568. IOS Press.

Wang, Z., Bapst, V., Heess, N., Mnih, V., Munos, R., Kavukcuoglu, K., and de Freitas, N. (2016). Sample efficient actor-critic with experience replay. *ArXiv e-prints*.

Weaver, L. and Tao, N. (2013). The optimal reward baseline for gradient-based reinforcement learning. *CoRR*, abs/1301.2315. URL `http://arxiv.org/abs/1301.2315`.

Wehenkel, A. and Louppe, G. (2019). Unconstrained monotonic neural networks. URL `https://arxiv.org/abs/1908.05164`.

Williams, R.J. (1992). Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine Learning*, 8(3), 229–256. doi:10.1007/BF00992696. URL `https://doi.org/10.1007/BF00992696`.