Policy gradient in Deep Reinforcement Learning

Ayoub Ghriss
Ecole Polytechnique
ayoub.ghriss@polytechnique.org
Van Huy Vo
Ecole Polytechnique
van-huy.vo@polytechnique.edu

Instructor: Alessandro Lazaric
ENS Cachan - Master MVA
alessandro.lazaric@inria.fr

1 Introduction

Reinforcement learning is a branch of machine learning that focuses on learning how an actor should behave in an environment (a policy) so as to maximize a notion of cumulative reward with applications in robotics, motor control,... Traditional approaches in reinforcement learning usually learns an optimal policy as the greedy policy of the optimal value function. However, these approaches suffer from severe limitations such as the lack of guarantees of the value function, the complexity arising from continuous state and action spaces,... and are inapplicable in many real problems. Policy gradient methods, on the other hand, propose to approximate the policy directly using generic function approximators, such as neural networks, which do not suffer from these limitations and have achieved great successes recently.

Some of state-of-the-art algorithms in this approach are REPS, TRPO and A3C. Although these algorithms are invented as heuristics to tackle different problems of policy gradient, in the paper [1], Neu et al. give arguments from the convex optimization perspective to show that they actually solve the same problem, only with different optimization algorithms and regularization functions. With this observation, the author also deduces some results on the convergence of these algorithms.

In this project, we aim to understand and explain the arguments in [1]. Furthermore, in the last part of the report, we present our implementation of the TRPO as an attempt to explore the performance and the limitations of such methods.

2 Policy gradient

2.1 Markov Decision Process and Average-Reward

In reinforcement learning, interactions between an actor and the environment is usually modeled as a Markov Decision Process (MDP) which is characterized by a state space $\mathcal{X}$ of all possible states, an action space $\mathcal{A}$ of all actions that the actor can perform, a transition probability $P(y|x,a) : \mathcal{X} \times \mathcal{A} \times \mathcal{X} \to \mathbb{R}$ reflecting the influence of taking an action over the access to different states, and a reward function $r(x,a) : \mathcal{X} \times \mathcal{A} \to \mathbb{R}$ showing the immediate reward for taking the action a while on the state x.

The decision making of the actor in this environment is characterized by a policy π. A policy in its most general form is a conditional probability distribution on the action space given a state, i.e $\pi(a|x)$ shows how likely the actor will perform the action a while on state x. We aim to learn the best policy with respect to some measure of goodness. In the course, we have seen the discounted cumulative reward as a measure to evaluate a policy. Neu et al. [1] consider another popular measure which is the average-reward. Given a policy π, its average-reward is defined as

$$\rho(\pi) = \lim_T \mathbb{E}\Big[\frac{1}{T}\sum_{i=1}^{T} r_t(X_t, A_t)\Big] \tag{1}$$

where $A_t \sim \pi(\cdot|X_t)$. In order to maximize the average-reward, we make the assumption that π induces a unique stationary state distribution ν_π over the state space that satisfies $\nu_\pi(y) = \sum_{x,a} P(y|x,a)\pi(a|x)\nu_\pi(x)$ for all $y \in \mathcal{X}$. The average reward then can be rewritten as

$$\rho(\pi) = \sum_{x,a} \nu_\pi(x)\pi(a|x)r(x,a) = \sum_{x,a} \mu(x,a)r(x,a). \tag{2}$$

where $\mu = \nu_\pi\pi$ is the stationary distribution over state-action space induced by π. Therefore, finding the optimal policy is equivalent to a Linear Programming problem on the state-action space. Denote by Δ the set of all feasible state-action distributions

$$\Delta = \{\mu : \mathcal{X} \times \mathcal{A} \to \mathbb{R} : \sum_b \mu(y,b) = \sum_{x,a} P(y|x,a)\mu(x,a)\ \forall y\}, \tag{3}$$

we aim to solve the optimization problem

$$\mu^* = \operatorname*{argmax}_{\mu\in\Delta} \rho(\mu) \tag{4}$$

where $\rho(\mu)$ is the average reward of the policy inducing μ. Note that we can recover the policy and the state distribution from μ by the relations $\nu(x) = \sum_a \mu(x,a)$ and $\pi(a|x) = \frac{\mu(x,a)}{\nu(x)}$. We also denote by ν_μ and π_μ the state distribution and the policy respectively defined as such.

As in the case of discounted cumulative reward, we also define the value function $V^\pi : \mathcal{X} \to \mathbb{R}$ as

$$V^\pi = \lim_{N\to\infty} \mathbb{E}\Big[\sum_{i=1}^{T} r_t(X_t, A_t) - \rho(\pi)|X_1 = x\Big] \tag{5}$$

which satisfies the equation

$$V^\pi(x) = \sum_{a\in\mathcal{A}} \mu_\pi(x,a)\Big[r(x,a) + \sum_y P(y|x,a)V^\pi(y) - \rho(\mu_\pi)\Big]. \tag{6}$$

The Q-function $Q^\pi : \mathcal{X} \times \mathcal{A} \to \mathbb{R}$ and the advantage function $A^\pi : \mathcal{X} \times \mathcal{A} \to \mathbb{R}$ are then defined as

$$Q^\pi(x,a) = r(x,a) + \sum_y P(y|x,a)V^\pi(y) - \rho(\mu_\pi) \tag{7}$$

and

$$A^\pi(x,a) = Q^\pi(x,a) - V^\pi(x). \tag{8}$$

Neu et al. also prove a relation between average-reward ρ and the advantage function by the following lemma

Lemma 1.

$$\rho(\pi') - \rho(\pi) = \sum_{x,a} \mu_{\pi'}(x,a)A^\pi(x,a). \tag{9}$$

2.2 Policy gradient

Solving the average-reward maximization (4) is a hard problem since the state-action space can be very large or even infinite and therefore testing all possible policies on this space is practically infeasible. To overcome this problem, policy gradient is a paradigm proposing to consider that the policy is a function on the state-action space parameterized by θ. The problem 4 then becomes finding θ^* such that

$$\theta^* = \underset{\theta}{\operatorname{argmax}}\ \rho(\pi_\theta) \tag{10}$$

With this formalism, we can use conventional gradient descent algorithms to find the optimal θ. The overall algorithm is then a multi-step algorithm. In each step, samples are collected by using the current policy then the gradient of the objective function w.r.t the parameter θ is approximated using these samples. Finally, the parameter θ is updated using the computed gradient to come up with a new policy. Similar to the case of discounted cumulative reward, Neu gives an equation to compute the gradient of ρ with respect to θ

Lemma 2.

$$\nabla\rho(\theta) = \sum_{x,a} \mu_{\pi_\theta}(x,a)\nabla \log \pi_\theta(a|x) A^\pi(x,a). \tag{11}$$

2.3 Regularized policy gradient

Doing vanilla policy gradient, however, has some major issues. Firstly, since the update only depends on new observations, the new policy can totally forget all old observations if the learning rate is too big. Secondly, since the updates are highly correlated, the policy will converge very slowly, if it does, to the optimal policy or it even may over-exploit the current policy and end up converging to a local optimum. To avoid these problems, it is desirable that the new policy stays close to the old policy while making as much as improvement as possible. This intuition are usually formulated in recent state-of-the-art algorithms by considering each update step as a constrained optimization problem.

Relative entropy policy search In [2], the authors proposed a way to control the size of policy update by imposing that the new policy should be close to the observed state-action distribution $q(x,a)$. This is formulated mathematically by upper bounding the Relative entropy of μ_π with respect to q

$$DL_{KL}(\mu_\pi \| q) \leq \delta. \tag{12}$$

Each policy update then consists in solving the optimization problem

$$\begin{aligned} &\mu_{\pi_{k+1}} = \underset{\mu\in\Delta}{\operatorname{argmax}}\ \rho(\mu) \\ &\text{subject to}\ \ DL_{KL}(\mu\|q) \leq \delta. \end{aligned} \tag{13}$$

The parameterized form of 13 is

$$\begin{aligned} &\theta_{k+1} = \underset{\theta}{\operatorname{argmax}}\ \rho(\mu_\theta) \\ &\text{subject to}\ \ DL_{KL}(\mu_\theta\|q) \leq \delta. \end{aligned} \tag{14}$$

Trust-Region Policy Optimization Similar to REPS, in [3], the authors control the size of the policy update while maximizing the improvement it brings by imposing that the new policy should be close to the old policy. This is formulated by uniformly upper bounding the total variation between the old policy and the new policy at all states

$$D_{TV}^{\max}(\pi_k\|\pi_{k+1}) = \max_x D_{TV}(\pi_k(a|x)\|\pi_{k+1}(a|x)) < \delta. \tag{15}$$

This upper bound on $D_{TV}^{\max}$ then replaced with $D_{KL}^{\max}$ by noticing that $(D_{TV}^{\max}(\pi_k\|\pi_{k+1}))^2 \leq D_{KL}^{\max}(\pi_k\|\pi_{k+1})$. Since this condition imposes a large number of constraints which make the problem difficult to solve in practice, the author finally propose to replace the upper bound on $D_{KL}^{\max}$ by the heuristic approximation on the average Kullback-Leibler divergence

$$D_{KL}^{\pi_k}(\pi_k\|\pi_{k+1}) = \mathbb{E}_{x\sim\nu_{\pi_k}}[D_{KL}(\pi_k(\cdot|x)\|\pi_{k+1}(\cdot|x))] \leq \delta. \tag{16}$$

Therefore, each policy update of TRPO consists in solving the optimization problem

$$\begin{aligned} \pi_{k+1} &= \underset{\pi}{\operatorname{argmax}} \sum_x \nu_{\pi_k}(x) \sum_a \pi(a|x) A^{\pi_k}(x,a) \\ &\text{subject to } \mathbb{E}_{x\sim\nu_{\pi_k}}[D_{KL}(\pi_k(\cdot|x)\|\pi(\cdot|x))] \leq \delta. \end{aligned} \tag{17}$$

The parameterized form of 17 is

$$\begin{aligned} \theta_{k+1} &= \underset{\theta}{\operatorname{argmax}} \sum_x \nu_{\pi_{\theta_k}}(x) \sum_a \pi_\theta(a|x) A^{\pi_{\theta_k}}(x,a) \\ &\text{subject to } \mathbb{E}_{x\sim\nu_{\pi_{\theta_k}}}[D_{KL}(\pi_{\theta_k}(\cdot|x)\|\pi_\theta(\cdot|x))] \leq \delta. \end{aligned} \tag{18}$$

Asynchronous Advantage Actor-Critic (A3C) Given in [5], A3C is an asynchronous version of advantage actor-critic algorithm. Although the main idea of the paper is introducing a paradigm in which multiple learners are trained simultaneously to guarantee that the sequence of samples during training is more stationary and to make the policy updates less correlated, thus makes the RL algorithms more robust, the authors found that in the case of applying this paradigm to advantage actor-critic algorithm, adding a constraint on the entropy of the policy encourages the policy to explore the environment and helps to avoid premature convergence to bad local optimums. Each update of A3C consists in firstly approximating the gradient of the surrogate objective

$$\rho(\mu_\theta) - \eta \sum_{\mathcal{X}} \nu_{\pi_k} \sum_{\mathcal{A}} \pi_\theta(a|x) \log \pi_\theta(a|x) \tag{19}$$

from sampled data then update the policy using normal gradient descent

$$\theta_{k+1} = \theta_k + \alpha \nabla_\theta \Big(\rho(\mu_\theta) - \eta \sum_{\mathcal{X}} \nu_{\pi_k} \sum_{\mathcal{A}} \pi_\theta(a|x) \log \pi_\theta(a|x) \Big). \tag{20}$$

Although A3C does not find the optimal parameter for the surrogate objective, it can be seen as attempting to optimize this function in each step.

These algorithms, while invented separately as heuristics to tackle different existing issues of policy gradient, can be unified under the point of view of convex optimization.

3 A unified view of REPS, TRPO and A3C

Neu et al. [1] gives arguments to show that from Convex Optimization viewpoint, REPS, TRPO and A3C actually solve the same convex optimization problem 4. They only differ in the choice of optimization algorithms as well as regularization functions. Inside this unified view, Neu et al. point out some interesting links between these algorithms and in particular deduces some results on their convergence. Neu et al. consider two optimization algorithms and two family of regularization functions.

3.1 Optimization algorithms

3.1.1 Mirror descent

Mirror descent is an algorithm to find the minimum of strictly convex functions over a convex set C. It can be seen as a special case of the generalized version of the projected gradient descent. In projected gradient decent, each update step is a combination of two stages. In the first stage, the gradient of the function with respect to the current parameter is computed then the parameter is adjusted making use of the gradient. After this stage, the updated parameter may no longer lie in C. Therefore, in the second step, the updated parameter is projected into C using Euclidean distance. Each step in projected gradient descent can be formulated as a optimization problem on its own:

$$x_{k+1} = \underset{x\in C}{\operatorname{argmin}} \Big(\langle x, \nabla f(x_k) \rangle + \alpha_k \|x - x_k\|^2 \Big) \tag{21}$$

In generalized projected gradient descent, we can change the Euclidean distance by an arbitrary function $d(x,y) : C \times C \to \mathbb{R}$ that satisfy

- $d(a, a) = 0$,
- $d(x, a) > 0$ for all $x \neq a$,
- $d(\cdot, a)$ is strictly convex and smooth for all a in the interior of C.

These functions are called positive-definite functions. The generalized projected gradient descent is then defined by the following iteration

$$x_{k+1} = \underset{x \in C}{\operatorname{argmin}} \left(\langle x, \nabla f(x_k) \rangle + \alpha_k d(x, x_k) \right) \tag{22}$$

where d is a positive-definite function. It is proved that with a proper choice of α_k, this algorithm effectively converges to the minimum point of f. In practice, d is usually chosen such that it facilitates the finding of optimal solution in each step.

A common class of semi-definite positive functions is the Bregmen divergence of strictly convex functions. Let ψ be a strictly convex function, its Bregman divergence is defined as

$$B_\psi(x, y) = \psi(x) - \psi(y) - \langle x - y, \nabla\psi(y) \rangle. \tag{23}$$

Since ψ is strictly convex, we can deduce immediately that $B_\psi(x, y)$ is positive-definite. Mirror descent is the generalized projected gradient descent when Bregman divergences are used in its iterations. Applying the mirror descent to solve 4, we have each step consists in solving the following optimization problem

$$\mu_{k+1} = \underset{\mu \in \Delta}{\operatorname{argmax}} \left(\rho(\mu) + \alpha_k B_\psi(\mu, \mu_k) \right) \tag{24}$$

noticing that since $\rho(\mu)$ is linear in μ, we have $\langle \mu, \nabla\rho(\mu_k) \rangle = \langle \mu, \nabla\rho(\mu) \rangle = \rho(\mu)$. In this context, we can also see Problem 24 as a regularized version of Problem 4.

3.1.2 Dual Averaging

Dual averaging is an optimization scheme that makes use of a proximal function $R : \mathbb{R}^d \to \mathbb{R}$ that is assumed to be strongly convex. Consider the optimization problem

$$\underset{x \in \mathcal{X}}{\text{minimize}} \ f(x)$$

where $f(x)$ is a smooth convex function. Dual averaging solves this optimization problem by generating a sequence of iterates (x_k, g_k). At the iteration k, Dual Averaging performs the update

$$g_{k+1} = g_k + \nabla f(x_k) \quad \text{and} \qquad x_{k+1} = \underset{x \in \mathcal{X}}{\operatorname{argmin}} (\langle g_{k+1}, x \rangle + \alpha_k R(x)) \tag{25}$$

where $(\alpha_k)_{k \geq 0}$ is a non-decreasing sequence. Note that the update step of x can be rewritten as

$$x_{k+1} = \underset{x \in \mathcal{X}}{\operatorname{argmin}} (\langle \frac{1}{k+1} \sum_{\tau=0}^{k} \nabla f(x_\tau), x \rangle + \frac{\alpha_k}{k+1} R(x)) \tag{26}$$

which explains the name of the algorithm. We can notice that this algorithm and the mirror descent algorithm above are similar since the update of x can be seen as a projection specified by R. Nevertheless, they are different in two ways: The first difference is that the gradient in mirror descent is evaluated only at the current value of x while the gradient in dual averaging is the average of all the gradient evaluated at the previous points. The second difference is that mirror descent requires the regularization term $d(x, x_k)$ to be the Bregman divergence of a strictly convex function while dual averaging requires the regularization term R to be strongly convex. However, in our problem, the average-reward ρ is linear in μ, therefore the two algorithms only differ in the regularization term.

Applying the dual averaging to solve Problem 4, we have each step consists in solving the following optimization problem

$$\mu_{k+1} = \underset{\mu \in \Delta}{\operatorname{argmax}} \left(\rho(\mu) + \alpha_k R(\mu) \right), \tag{27}$$

again noticing that $\rho(\mu)$ is linear in μ.

3.2 Regularization functions

Neu et al. [1] consider the negative Shannon entropy

$$R_S(\mu) = \sum_{x,a} \mu(x,a) \log \mu(x,a) \tag{28}$$

and the negative conditional entropy

$$R_C(\mu) = \sum_{x,a} \mu(x,a) \log \frac{\mu(x,a)}{\sum_b \mu(x,b)} = \sum_{x,a} \nu_\mu(x,a) \pi_\mu(a|x) \log \pi_\mu(a|x) \tag{29}$$

as well as their Bregman divergence $D_S(\mu\|\mu')$ and $D_C(\mu\|\mu')$ as regularization functions to feed in mirror descent and dual averaging algorithm. The explicit form of these two divergences are also given in [1] as

$$D_S(\mu\|\mu') = \sum_{x,a} \mu(x,a) \log \frac{\mu(x,a)}{\mu'(x,a)} \tag{30}$$

and

$$D_C(\mu) = \sum_{x,a} \mu(x,a) \log \frac{\pi(x,a)}{\pi'(x,a)} = \sum_x \nu_\pi(x) \sum_a \pi(a|x) \log \frac{\pi(x,a)}{\pi'(x,a)}. \tag{31}$$

It can be easily seen that $D_S(\mu\|\mu')$ is actually the Kullback-Leibler divergence $D_{KL}(\mu\|\mu')$ while $D_C(\mu\|\mu')$ is the expectation of the Kullback-Leibler divergence $D_{KL}(\pi\|\pi')$ over the state space $\mathcal{X}$. The author also proves that R_S and R_C are effectively strictly convex functions.

3.3 REPS, TRPO and A3C from the Convex Optimization perspective

With optimization algorithms and regularization functions above, Neu shows that REPS, TRPO and A3C actually attempt to solve Problem 4 by using a combination of these algorithms and regularizations.

REPS Neu et al. show that REPS is equivalent to solving Problem 4 using mirror descent with Bregman divergence D_S. Firstly, using Lagrangian strong duality, it can be shown that the hard constraint of Problem 13 can be incorporated to the objective function such that it is equivalent to

$$\mu_{\pi_{k+1}} = \operatorname*{argmax}_{\mu \in \Delta} \Big(\rho(\mu) - \eta D L_{KL}(\mu\|q) \Big) \tag{32}$$

for some $\eta > 0$. Since in each iteration, samples are drawn from the current policy, we have $q = \mu_k$ and $D_{KL}(\mu\|q) = D_S(\mu\|q) = D_S(\mu\|\mu_k)$. Therefore, we see that under this form, REPS is equivalent to the mirror descent algorithm using the Bregman divergence D_S. This equivalence allows the author to deduce that REPS converges to the optimal policy by simply using the convergence results of mirror descent algorithm. However, we see that this convergence result only applies for the exact form of REPS, i.e the policy π is not parameterized. In the case π is parameterized, we only have this convergence result if the class of functions represented by the parameterization contains the optimal policy.

TRPO the authors claim that TRPO is approximately equivalent to solving Problem 4 using mirror descent with Bregman divergence D_C. By examining [3], we see that replacing $D_{KL}(\pi_k(\cdot|x)\|\pi_{k+1}(\cdot|x))$ by $D_{KL}(\pi_{k+1}(\cdot|x)\|\pi_k(\cdot|x))$ does not change the arguments in the paper since the quantity $(D_{TV}^{\max}(\pi_{k+1}\|\pi_k))^2$ (which is symmetric) can be upper bounded by either $D_{KL}^{\max}(\pi_k\|\pi_{k+1})$ or $D_{KL}^{\max}(\pi_{k+1}\|\pi_k)$. The optimization problem 17 then can be replaced by

$$\begin{aligned} &\pi_{k+1} = \operatorname*{argmax}_{\pi} \ \sum_x \nu_{\pi_k}(x) \sum_a \pi(a|x) A^{\pi_k}(x,a) \\ &\text{subject to} \ \ \mathbb{E}_{x\sim\nu_{\pi_k}}[D_{KL}(\pi(\cdot|x)\|\pi_k(\cdot|x))] \le \delta. \end{aligned} \tag{33}$$

Now, similar to the argument for REPS above, we can incorporate the condition to the objective so that the problem 33 becomes

$$\begin{aligned}\pi_{k+1} &= \underset{\pi}{\operatorname{argmax}} \sum_x \nu_{\pi_k}(x) \sum_a \pi(a|x) A^{\pi_k}(x,a) - \eta \mathbb{E}_{x\sim\nu_{\pi_k}}[D_{KL}(\pi(\cdot|x)\|\pi_k(\cdot|x))] \\ &= \underset{\pi}{\operatorname{argmax}} \sum_x \nu_{\pi_k}(x) \sum_a \pi(a|x)\Big(A^{\pi_k}(x,a) - \eta \log \frac{\pi(x|a)}{\pi_k(x|a)}\Big).\end{aligned} \tag{34}$$

Now if we ignore the effect of changing the policy on the state distribution, i.e $\nu_{\pi_k} = \nu_{\pi_{k+1}}$ (this estimation is actually used in both variant *Single Path* and *Vine* of TRPO in [3], we see that the first sample s_0 is always drawn from ρ_0), by using Lemma 9, 34 becomes

$$\begin{aligned}\pi_{k+1} &= \underset{\pi}{\operatorname{argmax}} \sum_x \nu_{\pi}(x) \sum_a \pi(a|x)\Big(A^{\pi_k}(x,a) - \eta \log \frac{\pi(x|a)}{\pi_k(x|a)}\Big) \\ &= \underset{\pi}{\operatorname{argmax}} \Big(\rho(\pi) - \rho(\pi_k) - \eta \sum_x \nu_{\pi}(x) \sum_a \pi(a|x) \log \frac{\pi(x|a)}{\pi_k(x|a)}\Big) \\ &= \underset{\pi}{\operatorname{argmax}} \Big(\rho(\pi) - \eta D_C(\pi\|\pi_k)\Big)\end{aligned} \tag{35}$$

This form is actually the mirror descent algorithm using the Bregman divergence D_C for Problem 4. From the formulation 34, Neu et al. also gives a closed form update of the policy

$$\pi_{k+1}(x|a) \propto \pi_k(x|a) e^{\frac{A^{\pi_k}(x,a)}{\eta}} \tag{36}$$

and points out that this update is equivalent to the MDP Expert algorithm in [4] which is proven to minimize regret in an online setting, therefore TRPO also converges to the optimal policy. However, we notice again that this convergence result only applies for the exact form of TRPO and if the policy is parameterized, we do not necessarily have the convergence. Moreover, the result above is established only for the variants of TRPO that sampled from the current policy in each iteration. In [3], there are actually other variants of TRPO that work better in discrete tasks drawing, initially, actions from a uniform distribution.

A3C Using the update rule 20 of A3C, Neu et al. argue that each iteration of A3C actually attempt to optimize the surrogate objective 19 and if, again, we ignore the effect of changing the policy on the state distribution, the objective 19 can be rewritten as

$$\rho(\pi_\theta) - \eta \sum_{\mathcal{X}} \nu_{\pi_\theta}(x) \sum_{\mathcal{A}} \pi_\theta(a|x) \log \pi_\theta(a|x) = \rho(\mu_\theta) - \eta R_C(\mu_\theta). \tag{37}$$

Therefore, A3C is approximately equivalent to the dual averaging algorithm with the conditional entropy R_C for 4. However, since each policy update of A3C does not attempt to find the optimal π_θ for 37, we can not conclude about its convergence using known results on dual averaging. In order to have the convergence, Neu et al. propose to replace one single gradient descent in each step of A3C by solving the entire optimization problem

$$\text{maximizing} \quad \rho(\mu_\theta) - \eta R_C(\mu_\theta). \tag{38}$$

4 Implementation : Trust Region Policy Optimization

The TRPO has been introduced as an effective method to optimize large non-linear policies, such as neural networks, with robust performance. Hence, we will introduce TRPO in a more detailed manner, compared to previously mentioned methods, before discussing our implementation and experimental results.

We consider the problem of optimizing large nonlinear policies, such as neural networks, in an infinite-horizon discounted MDP $(\mathcal{X}, \mathcal{A}, P, r, \rho_0, \gamma)$, where $\gamma \in (0,1)$ is the discount factor.

We denote by $\pi : \mathcal{X} \times \mathcal{A} \to [0, 1]$ a stochastic policy, and $\eta(\pi)$ its expected discounted reward. Keeping the same previous notations for Q_π,V_π, and A_π, we have:

$$\eta(\pi) = \mathbb{E}_{s_0, a_0, \cdots \sim \pi} \left[\sum_{t=0}^{\infty} \gamma^t r(s_t) \right]$$
$$\text{where} \quad a_t \sim \pi(a_t|s_t), s_{t+1} \sim P(s_{t+1}|s_t, a_t) \text{ for } t \geq 0.$$

Let ρ_π be the discounted visitation frequencies

$$\rho_\pi(s) = P(s_0 = s) + \gamma P(s_1 = s) + \gamma^2 P(s_2 = s) + \dots,$$

where $s_0 \sim \rho_0$ and the actions are chosen according to π. The following identity expresses the expected return of another policy $\tilde{\pi}$ in terms of the advantage over π:

$$\eta(\tilde{\pi}) = \eta(\pi) + \sum_s \rho_{\tilde{\pi}}(s) \sum_a \tilde{\pi}(a|s) A_\pi(s, a). \tag{39}$$

4.1 TRPO formulation

We introduce the following local approximation to η (replacing ρ_π by $\rho_{\tilde{\pi}}$):

$$L_\pi(\tilde{\pi}) = \eta(\pi) + \sum_s \rho_\pi(s) \sum_a \tilde{\pi}(a|s) A_\pi(s, a). \tag{40}$$

We consider parameterized policies $\pi_\theta(a|s)$ with parameter vector θ, and we use the notation to use functions of θ rather than π: $\eta(\theta) := \eta(\pi_\theta)$, $L_\theta(\tilde{\theta}) := L_{\pi_\theta}(\pi_{\tilde{\theta}})$, and $D_{\mathrm{KL}}(\theta \parallel \tilde{\theta}) := D_{\mathrm{KL}}(\pi_\theta \parallel \pi_{\tilde{\theta}})$.

The core idea is that we can prove in our framework :

$$\eta(\theta) \geq L_{\theta_{\text{old}}}(\theta) - C D_{\mathrm{KL}}^{\max}(\theta_{\text{old}}, \theta)$$

with equality at $\theta = \theta_{\text{old}}$. Thus, by performing the following maximization, we are guaranteed to improve the true objective η:

$$\underset{\theta}{\text{maximize}} \left[L_{\theta_{\text{old}}}(\theta) - C D_{\mathrm{KL}}^{\max}(\theta_{\text{old}}, \theta) \right].$$

The advantage of TRPO is that it allows to take larger steps in a robust way by constraining the KL divergence between the new policy and the old policy:

$$\begin{aligned} &\underset{\theta}{\text{maximize}} \, L_{\theta_{\text{old}}}(\theta) \\ &\text{subject to } D_{\mathrm{KL}}^{\max}(\theta_{\text{old}}, \theta) \leq \delta. \end{aligned} \tag{41}$$

However, the constraint imposes that the KL divergence is bounded at every point in the state space, which is impractical to solve due to the large number of constraints. Instead, we can use a more relaxed heuristic approximation which considers the average KL divergence:

$$\overline{D}_{\mathrm{KL}}^{\rho}(\theta_1, \theta_2) := \mathbb{E}_{s \sim \rho} \left[D_{\mathrm{KL}}(\pi_{\theta_1}(\cdot|s) \parallel \pi_{\theta_2}(\cdot|s)) \right].$$

Therefore the TRPO proposes solving the following optimization problem to generate a policy update:

$$\begin{aligned} &\underset{\theta}{\text{maximize}} \, L_{\theta_{\text{old}}}(\theta) \\ &\text{subject to } \overline{D}_{\mathrm{KL}}^{\rho_{\theta_{\text{old}}}}(\theta_{\text{old}}, \theta) \leq \delta. \end{aligned} \tag{42}$$

4.2 Solving the constrained problem

4.2.1 Estimation of the Objective and Constraint

We use Monte Carlo simulation to approximate the objective and constraint functions. By expanding $L_{\theta_{\text{old}}}$ in Equation (42), we replace $\sum_s \rho_{\theta_{\text{old}}}(s) [\dots]$ in the objective by the

expectation $\frac{1}{1-\gamma}\mathbb{E}_{s\sim\rho_{\theta_{\text{old}}}}[\dots]$. Next, we replace $A_{\theta_{\text{old}}}$ by the Q-values $Q_{\theta_{\text{old}}}$ in Equation (**??**). Last, we replace the sum over the actions by an importance sampling estimator for the distribution q. TRPO problem in **??** is hence equivalent to the following one:

$$\begin{aligned} &\underset{\theta}{\text{maximize}}\, \mathbb{E}_{s\sim\rho_{\theta_{\text{old}}},a\sim q}\left[\frac{\pi_\theta(a|s)}{q(a|s)}Q_{\theta_{\text{old}}}(s,a)\right] \\ &\text{subject to } \mathbb{E}_{s\sim\rho_{\theta_{\text{old}}}}\left[D_{\text{KL}}(\pi_{\theta_{\text{old}}}(\cdot|s)\parallel\pi_\theta(\cdot|s))\right]\leq\delta. \end{aligned} \tag{43}$$

We then replace the expectations by sample averages and replace the Q value by an empirical estimate. We can then use two different schemes for performing this estimation:

- *single path*scheme, and is based on sampling individual trajectories.
- *vine*scheme, involves constructing a rollout set and then performing multiple actions from each state in the rollout set.

4.2.2 TRPO Algorithm

The method the article[3] described involves two steps:

- Compute a search direction, using a linear approximation to objective and quadratic approximation to the constraint
- perform a line search in that direction, ensuring that we improve the nonlinear objective while satisfying the nonlinear constraint.

We note by g_θ the gradient of the objective with respect to θ. We write the quadratic approximation of the constraint function :

$$\overline{D}_{\text{KL}}(\theta_{\text{old}},\theta)\approx\frac{1}{2}(\theta-\theta_{\text{old}})^T A(\theta-\theta_{\text{old}})$$

where $A_{ij}=\frac{\partial}{\partial\theta_i}\frac{\partial}{\partial\theta_j}\overline{D}_{\text{KL}}(\theta_{\text{old}},\theta)$.

Since A is symmetric semi-definite, we use the conjugate gradient method to solve $s=Ag_\theta$. The solution s is the search direction. We then obtain the maximal step scale $\beta=\sqrt{2\delta/s^TAs}$ that verifies $\delta=\overline{D}_{\text{KL}}\approx\frac{1}{2}(\beta s)^TA(\beta s)=\frac{1}{2}\beta^2s^TAs$ The last step is the line search, which allows to shrink the step exponentially until the objective improves.

The TRPO algorithm can be summarized as follows:

Algorithm 1 Trust Region Policy Optimization

Require: A parameterized policy π_θ and θ=θ_0
  for $i\leftarrow 1$ to T **do**
    Run policy for N trajectories
    Estimate advantage function at all time steps
    Compute objective gradient g_θ
    Compute A
    Use Conjugate Gradient to compute β and s
    Compute the rescaled update line search
    Apply update to θ
  end for

4.2.3 Implementation* and results

Policy choice We modeled the policy using a two-layers neural network : first fully connected layer of 64 neurons followed by a softmax classifier returning a vector associating a weight for each action. The states are flattened gray-scale images of size 64 by 64. Containing 262468 parameters, our version of the implementation is simpler than the neural policy stated by Schulman & Levine, but with a larger number of variables.

Results In contrast with the unified framework, the method in Schulman & Levine bounds $\overline{D}_{\mathrm{KL}}^{\rho_{\theta_{\mathrm{old}}}}(\theta_{\mathrm{old}}, \theta)$ instead of $\overline{D}_{\mathrm{KL}}^{\rho_{\theta_{\mathrm{old}}}}(\theta, \theta_{\mathrm{old}})$ with respect to θ. In our tests, we launched the training for few hours (compared to 30 hours in [3]). We have noticed, since $\overline{D}_{\mathrm{KL}}$ in not symmetric, that the original formulation of the TRPO converges to some locally optimal distributions. For instance, in the Breakout game, the player often ends up moving to the left side since the dot almost all the time heads to the left when the game starts. When using $\overline{D}_{\mathrm{KL}}^{\rho_{\theta_{\mathrm{old}}}}(\theta, \theta_{\mathrm{old}})$ instead, the player still explores other directions and ends up doing slightly better than the random policy after few hours.

While the effect of changing the variables order in the constraint seems to have an effect in our experiment, the large number of variables and the randomness of the decisions in our framework makes it difficult to conclude on this matter.

On the other hand, compared with the human performance in Atari Games as stated in [7], the gap is wider when the game presents random adversarial environment. In out tests in such type of games (Space Invaders), the algorithm needed several hours to perform slightly better than the uniform random policy. This might be mainly due to the form we've chosen for the neural policy, but it allowed to test some limitations of this method.

Conclusion

Policy gradient is a powerful paradigm to learn the optimal policy in many reinforcement learning problem especially those involving large or continuous state-action space. In this project, we have provided a general review of recent state-of-the-art algorithms in this paradigm such as REPS, TRPO and A3C. We have also explained Neu et al. arguments in [1] pointing out that from the convex optimization perspective, they actually solve the same problem and differ only in the choice of optimization algorithms as well as regularization functions. From this perspective, Neu et al. also deduce some results on the convergence of these algorithms. However, we observe that these results only apply in the exact form of these algorithms where the policy is not parameterized. In the other case, their convergence will much depend on the class of functions that the parameterization represents.

Therefore, the parameterization of the policy is a key element for a good performance of policy gradient methods. As we have seen with the TRPO results, choosing a suitable neural network structure for the policy makes the implementation of policy gradient methods more of an engineering task than a direct use of theoretical results.

References

[1] Gergely Neu & Vicenç Gómez & Anders Jonsson (2017) A Unified View of Entropy-Regularized Markov Decision Processes. *arXiv:1705.07798*.

[2] Jan Peters & Katharina Mulling & Yasemin Altun (2010) Relative Entropy Policy Search. *Proceedings of the Twenty-Fourth AAAI Conference on Artificial Intelligence (AAAI-10)*.

[3] John Schulman & Sergey Levine & Philipp Moritz & Michael Jordan (2015) Trust Region Policy Optimization. *arXiv:1502.05477*.

[4] Eyal Even-Dar & Sham. M. Kakade & Yishay Mansour (2009) On-line Markov Decision Processes. *Mathematics of Operations Research, 34(3):726–736, 2009*.

[5] Volodymyr Mnih & Adrià Puigdomènech Badia & Mehdi Mirza & Alex Graves & Timothy P. Lillicrap & David Silver & Koray Kavukcuoglu (2016) Asynchronous Methods for Deep Reinforcement Learning. *arXiv:1602.01783*.

[6] Lin Xiao (2010) Dual Averaging Methods for Regularized Stochastic Learning and Online Optimization. *Journal of Machine Learning Research 11 (2010) 2543-2596*.

[7] Mnih, V., Kavukcuoglu, K., Silver, D., Graves, A., Antonoglou, I., Wierstra, D., and Riedmiller, M. Playing Atari with deep reinforcement learning. *arXiv:1312.5602, 2013.*