INITIATIVE FOR APPLIED ARTIFICIAL INTELLIGENCE

Natural, Trust Region and Proximal Policy Optimization

MICHAEL PANCHENKO, appliedAI

August 10th, 2021

We present an overview of the theory behind three popular and related algorithms for gradient based policy optimization: natural policy gradient descent, trust region policy optimization (TRPO) and proximal policy optimization (PPO). After reviewing some useful and well-established concepts from mathematical optimization theory, the algorithms can be introduced in a very unifying manner.

Contents

Mathematical optimization (convex and non-convex) is a large and mature field of research, as is policy optimization in reinforcement learning. A fusion of ideas from the two fields gives rise to improvements upon naive gradient descent as used in policy gradient algorithms like REINFORCE. The improvements considered in the present text boil down to optimizing a regularized or constrained objective function rather than just the discounted expected sum of rewards. While trust-region policy optimization (TRPO) essentially uses natural gradients and restricts the step size to enforce proximity constraints on policy updates, the significantly simpler proximal policy optimization (PPO) introduces constraints that modify the original objective in a more RL-specific fashion. The actual practical algorithms are motivated by rigorous analysis but involve several approximations and simplifications. Experimental results of TRPO and PPO display much more stability and reliability than their non-regularized counterparts and thereby show that many desired properties are not lost in the approximations. PPO is a state-of-the-art algorithm which is implemented in most major RL frameworks, like RLLIB, STABLE-BASELINES and others.

1 Trust region methods

Before we dive into the main results of the papers, let us briefly describe trust region methods and other general methods for numerical optimization.[1] As we will see, several main results and improvements in the TRPO and PPO papers are very natural and follow standard approaches from the point of view of optimization.

In this section we will be concerned with minimizing a twice differentiable function $f : \mathbb{R}^N \to \mathbb{R}$. First notice that several optimization algorithms can be understood as the successive application of the following steps:

[1] For a self-contained introduction, we recommend [NW06].

Algorithm 1
Approximation Based Optimization
Select some x_0 as starting point. At each iteration k:

1. Find a local approximation to f close to the current point x_k that can be minimized (over the entire domain of definition $\mathcal{D}_x$) with less effort than f itself. Such approximations are often called *models*, and we will denote them by m_k. Find a minimum of the model.[2]
$$x_{\min} \approx x^* \in \operatorname{argmin}_{x \in \mathcal{D}_x} (m_k(x)).$$
2. Set $x_{k+1} \longleftarrow x_{\min}$ and repeat.

[2] Note how here we allow for m_k without unique minima, as is the case in applications in RL. However, in many applications m_k will be convex and argmin m_k will be a single point. For gradient based optimization methods this distinction is usually of little practical importance - one uses the point where gradient descent has converged.

For example, the following quadratic approximation to f:

$$\begin{aligned} m_k^{\text{grad}}(x) \;:=\; & f(x_k) + (x - x_k)^\top \cdot \nabla f(x_k) \\ & + \frac{1}{2t_k} \| x - x_k \|^2 \end{aligned}$$

yields the update rule of gradient descent with step size t_k:

$$x_{k+1} = x_k - t_k \nabla f(x_k).$$

For small enough t_k and f with Lipshitz continuous gradients, the above model actually forms a majorant of f (many proofs of convergence of gradient descent are based on this). However, the above approximation is clearly very crude and does not take into account the curvature of f. Constructing the model with the second order expansion of f results in Newton's Method. With the model

$$\begin{aligned} m_k^{\text{Nwt}}(x) \;:=\; & f(x_k) + (x - x_k)^\top \cdot \nabla f(x_k) \\ & + \frac{1}{2}(x - x_k)^\top H_k (x - x_k), \end{aligned} \tag{1}$$

where H_k is the Hessian of f at x_k, we obtain the update rule:

$$x_{k+1} = x_k - H_k^{-1} \nabla f(x_k).$$

The strategy outlined above clearly has a flaw: the models will almost always be poor approximations to f far from x_k - something that we have not taken into account. This situation can be improved by restricting the parameter region where we trust our model prior to optimization - hence the name *trust region*.

Usually, we cannot judge a priori whether the model is sufficiently good within the selected region (if it is not, then the region was too large) or whether on the contrary the region can be extended to allow for larger step sizes. A posteriori, however, we can compare the actual improvement in the objective, Δf to the improvement in the model Δm (the *predicted improvement*) . If the ratio $\Delta f / \Delta m$ is too small, we should reject the step and shrink the trust region. If the ratio is too large, we should increase it. To summarize, the improved algorithm works as follows [CGT00]:

Algorithm 2
Trust Region Optimization

Select some x_0 as starting point. At each iteration k:

1. Choose an approximation of f around x_k, call it $m_k\colon \mathbb{R}^N \longrightarrow \mathbb{R}$.
2. Choose a trust region U_k that contains x_k. Usually
$$U_k := \{x\colon \|x - x_k\|_k \leq \Delta_k\},$$
for some norm $\|\cdot\|_k$ and radius $\Delta_k > 0$.
3. Find an approximation of a minimum of m_k within the trust region
$$x_{\min} \approx x^* \in \operatorname{argmin}_{x \in U_k} m_k(x).$$
4. Compute the ratio of *actual-to-predicted improvement*:
$$\rho_k := \frac{f(x_k) - f(x_{\min})}{m_k(x_k) - m_k(x_{\min})}.$$
If ρ_k is sufficiently large, accept $x_{\min}$ as the next point (i.e. $x_{k+1} \leftarrow x_{\min}$), otherwise don't move ($x_{k+1} \leftarrow x_k$)
5. If ρ_k is sufficiently large, increase the next trust region, if it is too small then shrink it. With U_k as above, this step usually amounts to scaling the radius, i.e. $\Delta_{k+1} = \varepsilon_k \Delta_k$ with the scaling factor ε_k depending on ρ_k.
6. Increment k by 1 and go to step 1.

Figure 1 illustrates this scheme with the contour lines of a quadratic approximation of the true objective shown at several steps.

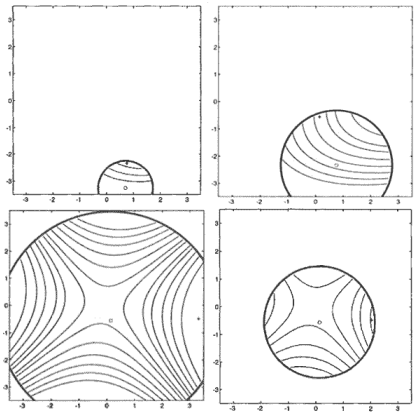

Figure 1. Trust Region Optimization. Four steps of a trust region algorithm with a quadratic model; from left to right, top to bottom. The true objective is not shown. In the first step ρ_0 is sufficiently large, so $x_{\min}$ is accepted and the trust region increased. Same for the second step. In the third step the trust region is too large, the model is not a good approximation of the objective and as consequence ρ_2 is not large enough. Therefore, the update of x_2 is rejected and instead the trust region is decreased, giving a different $x_{\min}$, which now can be accepted. Images taken from [CGT00].

2 Update rules of trust region methods

Let us ignore steps 4 and 5 from the above algorithm for a moment and analyze what kind of update rules we obtain from different trust region constraints. With a linear model

$$m_{x_k}^{\text{linear}}(x) = f(x_k) + (x - x_k)^\top \cdot \nabla f(x_k), \tag{2}$$

and a trust region $U_k := \left\{x\colon \frac{1}{2}\|x - x_k\|^2 \leq \delta^2\right\}$ we have the update rule

$$x_{k+1} = x_k - \delta \frac{\nabla f(x_k)}{\|\nabla f(x_k)\|}.$$

This update is commonly known as *normalized gradient descent*.

Using the quadratic approximation (1) as model with the same constraint as above we obtain the rule:

$$x_{k+1} = x_k - (H_k + \lambda)^{-1} \nabla f(x_k),$$

where λ is the Lagrange multiplier of the constraint. This update rule is the essence of the widely used *Levenberg-Marquardt algorithm* [Mar63].

A Euclidian sphere in the parameter space $\mathbb{R}^N$ typically does not reflect the geometry relevant for the optimization problem. Some problem-specific distance on $\mathbb{R}^N$ might do a better job in specifying what it means for x to be too far away from x_k. A particularly simple and useful choice is often a suitable symmetric and positive-definite matrix $F(x_k)$, which could be e.g. a metric tensor induced by some distance function. The trust region constraint then becomes

$$\frac{1}{2}(x - x_k)^\top F(x_k)(x - x_k) \leq \delta^2, \tag{3}$$

and the update rule from the linear model (2) takes the form

$$x_{k+1} = x_k - D_k \nabla f(x_k), \tag{4}$$

where

$$D_k := \frac{\delta}{\sqrt{\nabla f^\top(x_k) F^{-1}(x_k) \nabla f(x_k)}} F^{-1}(x_k).$$

An interesting choice for F is the Hessian H_k of f at x_k, which essentially means that distances between x and x_k are measured in units of the curvature of f at x_k. Then the update rule (4) corresponds to a variant of the *damped Newton method*. As an alternative for the full Hessian, one could use only the diagonal components of H_k for setting up the constraint, i.e. $F_k := \mathrm{diag}(H_k)$ (a choice that is supported by heuristics). Then, using a trust region of the form (3) together with the second order model (1) results in *Fletcher's modification* of the Levenberg-Marquardt update rule [Fle71]:

$$x_{k+1} = x_k - [H_k + \lambda \,\mathrm{diag}\,(H_k)]^{-1} \nabla f(x_k).$$

Using linear or quadratic approximations with different trust region constraints gives rise to many of the standard optimization algorithms that are commonly used in practice. As we will see below, several improvements of the naive policy gradient algorithm in reinforcement learning are based on exactly the same ideas that we highlighted in this section.

3 Notation and the RL objective

In all that follows we will consider an infinite horizon Markov decision process with states s_t and actions a_t. For notational simplicity, we assume that state and action spaces are discrete.[3] Here we are in the context of model-free reinforcement learning and we want to find a policy from a family of functions $\{\pi_\theta : \theta \in \mathbb{R}^n\}$ differentiable wrt. their parameters. The goal is to find the θ maximizing the **expected sum of discounted rewards**

$$\eta_{\pi_\theta} = \mathbb{E}_{\pi_\theta}\left[\sum_{t=0}^{\infty} \gamma^t r(s_t, a_t)\right]. \tag{5}$$

In what follows, we will often omit the subscript θ when it is clear which policy is meant, or if a statement applies to any policy π. We write π_k for π_{θ_k}.

[3] With some effort, the results stated can be transferred to continuous spaces.

We assume stochastic transition dynamics and, for notational simplicity, denote by τ_t the conditional distribution of $s_{t+1} \mid a_t, s_t$.[4] The value, Q and advantage functions are defined as

$$\begin{aligned} V_\pi(s_t) &= \mathbb{E}_\pi\left[\sum_{k=0}^{\infty} \gamma^k r(s_{t+k})\right] \\ &= r(s_t) + \gamma\, \mathbb{E}_{a_t \sim \pi(s_t)}[\mathbb{E}_{\tau_t}[V_\pi(s_{t+1})]], \end{aligned}$$

$$Q_\pi(s_t, a_t) = r(s_t) + \gamma\, \mathbb{E}_{\tau_t}[V_\pi(s_{t+1})],$$

and

$$A_\pi(s_t, a_t) = Q_\pi(s_t, a_t) - V_\pi(s_t).$$

[4] This is not to be confused with the usual notation of τ for the distribution of trajectories.

From these definitions directly follows:

$$\begin{aligned} V_\pi(s_t) &= \mathbb{E}_{a_t \sim \pi(s_t)}[Q(s_t, a_t)], \\ 0 &= \mathbb{E}_{a_t \sim \pi(s_t)}[A_\pi(s_t, a_t)], \end{aligned} \tag{6}$$

and

$$\begin{aligned} A_\pi(s_t, a_t) &= r(s_t) + \gamma\, \mathbb{E}_{\tau_t}[V_\pi(s_{t+1})] \\ &\quad - V_\pi(s_t). \end{aligned} \tag{7}$$

3.1 Approximating the objective

As explained in Section 1, before we start the optimization at each step k we need a local model around θ_k for the full objective (5).[5] We will use the following approximation to η_π:

$$m_k(\theta) := \eta_{\pi_k} + \sum_s \rho_{\pi_k}(s)\, \mathbb{E}_{a \sim \pi_k(s)}\left[\frac{\pi_\theta(a \mid s)}{\pi_k(a \mid s)} A_{\pi_k}(a, s)\right], \tag{8}$$

[5] In the literature referenced, the notation θ_{old} is used instead of θ_k.

where

$$\rho_\pi(s) := \sum_{t=0}^{\infty} \gamma^t \, \mathbb{P}_\pi(s_t = s) \tag{9}$$

is the (un-normalized) **discounted state density**, with $s_0 \sim \rho_0$ for some initial state distribution ρ_0.[6] Due to the vanishing expectation of the advantage function (6), we immediately see that

$$m_k(\theta_k) = \eta_{\pi_k}.$$

Moreover:

$$\nabla m_k(\theta_k) = \mathbb{E}_{\pi_\theta}[\nabla \log \pi_{\theta_k}(a \mid s)\, Q_{\pi_k}(a, s)],$$

which is a well-known expression for the policy gradient [SMSM99]. Therefore, (8) is a first order approximation to η_{π_θ} around θ_k.

With a slight abuse of notation, in modern literature (including the PPO paper [SWD+17]) equation (8) is usually written as

$$m_k(\theta) := \eta_{\pi_k} + \mathbb{E}_{\pi_k}\left[\frac{\pi_\theta(a \mid s)}{\pi_k(a \mid s)} A_{\pi_k}(a, s)\right].$$

4 Natural gradients, TRPO and PPO

With the very basic definitions that we have set up above, we can already make a lightning summary of the results of three important papers in RL. All one has to do is to combine ideas from trust region optimization with the objective/model (8).

Since policies are probability distributions, a natural notion of distance for them is given by the KL-divergence.[7] Even though it is not symmetric and therefore cannot really be understood as a proper distance, its second order approximation is the positive definite Fisher information matrix which can be used as a metric tensor for constructing the constraint

$$D_{\mathrm{KL}}(\pi_k(\cdot \mid s), \pi_\theta(\cdot \mid s)) \approx \frac{1}{2}(\theta - \theta_k)^\top F(\theta_k, s)\,(\theta - \theta_k),$$

with

$$F(\theta_k, s)_{i,j} = \frac{\partial^2}{\partial\theta_i \partial\theta_j} D_{\mathrm{KL}}(\pi_k(\cdot \mid s), \pi_\theta(\cdot \mid s)).$$

Thus, a natural bound for π_θ not being too far from π_k is to demand that on average the second order approximation of the KL-divergence of π_θ and π_k is smaller than some radius δ, resulting in the trust region constraint

$$\frac{1}{2}(\theta - \theta_k)^\top \mathbb{E}_{\pi_k}[F(\theta_k, s)]\,(\theta - \theta_k) \le \delta. \tag{10}$$

[6] If there is a discount factor $\gamma<1$, the state density is not a probability distribution since $\sum_s \rho_\pi(s) = \frac{1}{1-\gamma}$. If no discount factor is used, ρ_π usually denotes the stationary distribution of states after following π for a long time, which is assumed to exist. I.e. $\rho_\pi(s) := \lim_{T\to\infty} \frac{1}{T}\sum_{t=0}^{T} \mathbb{P}_\pi(s_t = s)$. This is a proper probability distribution. In the RL literature one can often find expressions of the type $\mathbb{E}_{\rho_\pi(s)}[f(s)]$, which should be interpreted as $\sum_s \rho_\pi(s) f(s)$, even when a discount factor is present and ρ_π is not properly normalized.

[7] One reason for the "naturality" of KL-divergence for constraints is its information-theoretic interpretation. Note that it also forms an upper bound of the total variation divergence δ through Pinsker's inequality $\delta(P, Q) \le \sqrt{\frac{1}{2} D_{\mathrm{KL}}(P \| Q)}$, where P, Q are arbitrary probability distributions. However, arguably the main reason that makes it natural is that its second derivative (for distributions from a parametrized function family), the Fisher information matrix that we use below to formulate trust-region constraints, gives a metric tensor on the corresponding space of probability distributions. This means that distances between policies measured according to the Fisher information are parametrization invariant. This line of thought was used in the paper that originally introduced natural gradients for RL [Kak01].

This is a constraint of the type (3) which hence results in the following instantiation of the update rule (4):[8]

$$\theta = \theta_k - D_k \nabla m_k(\theta_k), \tag{11}$$

with

$$D_k := \frac{\delta}{\sqrt{\nabla m_k^T(\theta_k)\, \bar{F}^{-1}(\theta_k) \nabla m_k(\theta_k)}} \bar{F}^{-1},$$

where $\bar{F}$ is the average Fisher information:

$$\bar{F}(\theta) := \mathbb{E}_{\pi_\theta}[F(\theta, s)]. \tag{12}$$

[8] Originally, the trust region approach based on (an approximation of) the KL-divergence was motivated in a different way. One can prove that optimizing a certain surrogate objective with the KL-divergence as penalty results in guaranteed monotonous improvement in the expected sum of rewards. See Section 5 for more details.

4.1 Natural Gradient and TRPO

In the paper that introduced the natural gradient for RL [Kak01], the above update rule was implemented with a fixed step size multiplying the update direction $\bar{F}^{-1}(\theta_k) \cdot \nabla m_k(\theta_k)$ instead of the adaptive step size coming from the constraint. This choice was already an improvement over the naive policy gradient. The only practical difference between the method introduced in the trust region policy optimization paper [SLA+15] and natural gradient is that the constraint (10) is taken more strictly in TRPO (it will actually be fulfilled with each update) and the step size is not fixed.

The reader might have noticed that in all considerations above we kept silently ignoring a crucial element of the general trust region optimization algorithm - namely the possibility to reject an update and to adjust the trust region based on differences between predicted and actual improvement, steps 4 and 5 in algorithm (2). Indeed, the natural gradient completely foregoes these steps, which is precisely why it performs worse than TRPO. However, what about TRPO itself - is there any rejection of points and region adjustment happening? In fact, nothing of the like is mentioned in the paper's main text. However, in Appendix C, where the implementation details are described, it is mentioned that a backtracking line search is performed in the direction $\bar{F}^{-1}(\theta_k) \cdot \nabla m_k(\theta_k)$, starting from the maximal step size $\frac{\delta}{\sqrt{\nabla m_k \bar{F}^{-1} \nabla m_k}}$, until a sufficient increase in the objective is observed. This procedure can be understood as a way of rejecting updates and reducing the trust region if the true increase is too small. The authors also say that without such a line search, from time to time the algorithm would take too large steps with catastrophic drops in performance. Such behavior is known to happen in trust region algorithms where the region is never adjusted, see [CGT00]. Although part of the general specification of a trust region algorithm is still not applied in TRPO - namely

comparing predicted and actual improvements and possibly extending the trust region, not just shrinking it, by following algorithm (2) more closely than the fixed-size natural gradient updates, the authors of TRPO introduced a very popular and robust state-of-the-art (at that time) policy optimization algorithm.[9]

[9] From the point of view presented here, it seems natural to wonder whether by following the general trust region approach even more closely and adjusting the trust region based on the same criteria as described in it, the performance would increase even more (see chapter 6 in [CGT00] for a slightly stricter but less general specification of the trust region algorithm).

4.2 Proximal Policy Optimization

The observation that trust region extension and shrinking were not fully applied in TRPO but some version of them was still needed to prevent catastrophic collapse brings us to the next improvement upon TRPO - proximal policy optimization (PPO) [SWD+17]. The main idea here is the following: all we really care about is optimizing the model (8) without taking too large steps in policy space. While using the KL divergence is a natural choice for bounding deviations of general probability distributions, one could make an even more suitable choice that is specific to the actual model of the objective. In the PPO paper this choice can be phrased as following: the new policy is only allowed to improve upon the old one by staying close to it in a pointwise, objective-specific sense, specified in equation (13) below. It is, however, allowed to make large steps in θ which decrease the expected sum of rewards. This approach is different from the KL-based constraint which is indifferent to whether deviations in a policy lead to an improvement or not. Moreover, the KL-based constraint is based on differences of policies as a whole (as probability distributions), allowing for large pointwise differences.

Denoting

$$r_\theta(a,s) := \frac{\pi_\theta(a \mid s)}{\pi_k(a \mid s)},$$

the quantity that is to be maximized at the k-th iteration is:

$$\mathbb{E}_{\pi_k}[r_\theta(a,s)\, A_{\pi_k}(a,s)].$$

The PPO idea for restricting steps results in the following recipe, for some (small) $\epsilon > 0$ that is chosen as hyperparameter:

- If $A_{\pi_k}(a,s) > 0$, then we don't allow $r_\theta(a,s)$ to become larger than $1+\epsilon$.
- If $A_{\pi_k}(a,s) < 0$, then $r_\theta(a,s)$ should not become smaller then $1-\epsilon$.

A very similar behavior can also be implemented as trust-region constraints on θ of the form

$$\operatorname{sign}(A_{\pi_k}(a,s))\,(r_\theta(a,s)-1) \leq \epsilon, \tag{13}$$

where we would have one constraint for each (a, s). However, in the PPO paper these conditions were not included as constraints but rather the objective itself was modified to

$$m_k^{\mathrm{PPO}}(\theta) := \eta_{\pi_k} + \mathbb{E}_{\pi_k,t}[\min(r_\theta A_{\pi_k}, \mathrm{clip}(r_\theta A_{\pi_k}))], \tag{14}$$

where the clip function clips r to $1 \pm \epsilon$ based on the sign of A_{π_k} if it is too large or too small. To first order this coincides with the previous model (8) used in natural gradients and TRPO. Note that improvements to the objective due to policies too far from π_k are cut off, meaning that $m_k^{\mathrm{PPO}}(\theta) \leqslant m_k(\theta)$. Therefore, the clipped objective can be understood as a minorant for the unclipped one. Thus, schematically PPO algorithms take the form:

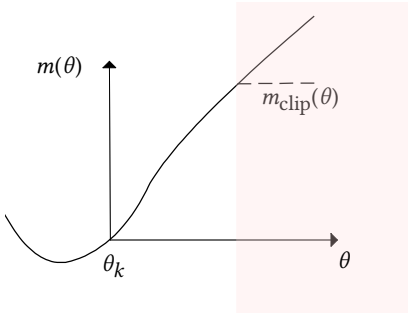

Figure 2. Constraining vs. clipping. Maximizing m w.r.t. the constraint (highlighted region) is almost equivalent to maximizing the clipped objective m_{clip} without any constraints by gradient methods, starting from θ_k and making small steps. As soon as θ moves into the constrained region, $\nabla m_{\mathrm{clip}}(\theta)$ vanishes, so e.g. for gradient ascent with step size α, one would violate the constraint by at most $\alpha \nabla m(\theta^*)$, with θ^* being the penultimate value before convergence of gradient ascent (i.e. sufficiently close to the constraint boundary).

Algorithm 3
PPO-style optimization

1. Unroll the current policy π_k in order to estimate

   $$\mathbb{E}_{\pi_k,t}[\min(r_\theta A_{\pi_k}, \mathrm{clip}(r_\theta A_{\pi_k}))]$$

   This may involve batching and different strategies for estimating advantages.

2. Run gradient ascent in θ in order to maximize the objective obtained in the previous step, thereby obtaining $\theta_{\max}$.

3. Set $\theta_k \leftarrow \theta_{\max}$ and repeat from step 1.

From a practical point of view, optimizing the original objective (8) with a sufficiently expressive policy π_θ and gradient methods while satisfying the constraints (13) is essentially equivalent to optimizing the clipped objective (14), meaning that PPO can be viewed as a trust-region optimization as well. Figure 2 can be helpful to understand this statement.

Optimizing the clipped objective will occasionally lead to slight violations of the constraints. This is because the clipped objective only discourages going into the constrained region and does not strictly enforce it. One can view the clipping strategy as a computationally efficient way to deal with constraints of the type (13).[10]

PPO is much simpler than TRPO both conceptually and in implementation, while often equaling or outperforming the latter. Intuitively, this might be due to the much more RL-specific constraints (13) on the steps in θ, compared to the general-purpose KL-divergence based constraint (10). Again, as in previous algorithms, adjustments to the trust region based on predicted vs. actual improvements do not form part of the PPO algorithm and one might reasonably speculate that adjusting ϵ based on such criteria could lead to further improvements of performance.

[10] To the author's knowledge, the point of view that clipping the objective for gradient-based optimization is essentially equivalent to imposing trust-region constraints, albeit not the same ones as in the TRPO paper, has not been mentioned in the literature. This perspective allows for a more straightforward comparison of PPO and other approaches.

5 TRPO from monotonous improvement

The derivation of the TRPO update rule (11) in the original paper [SLA+15] was based on quite a different line of thought from the one presented in Section 4 above. The constraint in average Fisher information (10) was motivated by a proof of monotonic improvement of η_π from optimization of a surrogate objective where the KL-divergence is used as penalty. We will give a brief overview of the ideas and calculations that form the backbone of the original paper.

The following useful identity expresses the rewards of one policy $\tilde{\pi}$ in terms of expectation of advantages of another policy π:

$$\eta_{\tilde{\pi}} = \eta_\pi + \mathbb{E}_{\tilde{\pi}}\left[\sum_{t=0}^{\infty} \gamma^t A_\pi(s_t, a_t)\right]. \tag{15}$$

This can be easily proved by observing that

$$\mathbb{E}_{s_t, a_t \sim \tilde{\pi}} \mathbb{E}_{\tau_t}[V_\pi(s_{t+1})] = \mathbb{E}_{\tilde{\pi}}[V_\pi(s_{t+1})],$$

and using the telescopic nature of the sum $\sum_{t=0}^{\infty} \gamma^t A_\pi(s_t, a_t)$. After some computation, equation (15) can be rewritten in state space as:

$$\eta_{\tilde{\pi}} = \eta_\pi + \sum_s \rho_{\tilde{\pi}}(s)\, \mathbb{E}_{a \sim \tilde{\pi}(s)}[A_\pi(s, a)].$$

One might expect that the state distribution ρ_π does not change too quickly when π is smoothly varied, since even if different actions are taken, the unchanged system dynamics should smooth out the behavior of ρ_π. This intuition is useful for finding good approximations: replacing $\rho_{\tilde{\pi}}$ by ρ_π in the above expression gives a first order approximation to $\eta_{\tilde{\pi}}$, namely exactly the model $m_\pi(\tilde{\pi})$ (8) that we have already used above (where we have also shown that it is truly a first order approximation).[11] Now one would hope that finding a new policy π that improves the approximation given by m_π could guarantee an improvement in $\eta_{\tilde{\pi}}$, as long as the improved policy π stays within a sufficiently small region around $\tilde{\pi}$. Since the KL-divergence gives a way of measuring proximity of distributions, it is maybe not too surprising that such a bound can be derived using it. One of the central results of the TRPO paper is the following relation:

[11] In the TRPO paper this model is called L_π.

$$\eta_{\tilde{\pi}} \geq m_\pi(\tilde{\pi}) - C D_{\mathrm{KL}}^{\max}(\pi, \tilde{\pi}),$$

where C is a constant depending on π and γ and

$$D_{\mathrm{KL}}^{\max}(\pi, \tilde{\pi}) = \max_s [D_{\mathrm{KL}}(\pi(\cdot \mid s), \tilde{\pi}(\cdot \mid s))].$$

Thus, optimizing $m_\pi(\tilde{\pi}) - C D_{\mathrm{KL}}^{\max}(\pi, \tilde{\pi})$ w.r.t. $\tilde{\pi}$ leads to guaranteed improvement in the objective. The term $C D_{\mathrm{KL}}^{\max}(\pi, \tilde{\pi})$ acts as a penalty for moving $\tilde{\pi}$ too far from π. Now, in practice this penalty leads to too small steps and $D_{\mathrm{KL}}^{\max}$ is difficult to estimate. Therefore, the authors propose to replace $D_{\mathrm{KL}}^{\max}$ by the average KL-divergence and to incorporate the requirement to stay close to π in KL-divergence as constraint rather than as penalty. This argumentation leads to the trust-region algorithm that we presented in Section 4.1.

6 Conclusion

The TRPO and PPO papers achieved significant improvements to existing RL optimization methods. The core ideas in both can be summarized as: optimize the discounted sum of rewards through steps that don't bring you *too far away* from the previous policy by either enforcing constraints or modifying the objective in a smart way.

While naive policy gradient formulates "too far away" as keeping the update in parameter space $\Delta\theta = \theta - \theta_k$ small, in TRPO "too far away" is formulated in a more natural sense for maps from states to probability distributions, i.e. using an approximation to average KL divergence, (12). In PPO "too far away" is defined in an intrinsically RL-like way: by not allowing the approximation to improve the expected sum of advantages through point-wise constraints on $\pi(a \mid s)$ over all states and actions (equations (13) and (14)). Many of these ideas seem very natural from the point of view of optimization theory [Nes04], especially of trust region methods [CGT00, NW06].

BIBLIOGRAPHY

[CGT00] Andrew R. Conn, Nicholas I. M. Gould, and Philippe L. Toint. *Trust Region Methods.* Society for Industrial and Applied Mathematics, jan 2000.

[Fle71] R Fletcher. A modified Marquardt subroutine for nonlinear least squares. Technical Report, Atomic Energy Research Establishment, Harwell, Berskshire, 1971.

[Kak01] Sham M. Kakade. A Natural Policy Gradient. *Advances in Neural Information Processing Systems*, 14, 2001.

[Mar63] Donald W. Marquardt. An Algorithm for Least-Squares Estimation of Nonlinear Parameters. *Journal of the Society for Industrial and Applied Mathematics*, 11(2):431–441, jun 1963.

[Nes04] Yurii Nesterov. Nonlinear Optimization. In Yurii Nesterov, editor, *Introductory Lectures on Convex Optimization: A Basic Course*, Applied Optimization, pages 1–50. Springer US, Boston, MA, 2004.

[NW06] Jorge Nocedal and S. Wright. *Numerical Optimization.* Springer Series in Operations Research and Financial Engineering. Springer-Verlag, New York, Second edition, 2006.

[SLA+15] John Schulman, Sergey Levine, Pieter Abbeel, Michael Jordan, and Philipp Moritz. Trust Region Policy Optimization. In *International Conference on Machine Learning*, pages 1889–1897. PMLR, jun 2015.

[SMSM99] Richard S. Sutton, David McAllester, Satinder Singh, and Yishay Mansour. Policy Gradient Methods for Reinforcement Learning with Function Approximation. *Advances in Neural Information Processing Systems*, 12, 1999.

[SWD+17] John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal Policy Optimization Algorithms. *ArXiv:1707.06347 [cs]*, aug 2017.