MITSUBISHI ELECTRIC RESEARCH LABORATORIES
https://www.merl.com

Quasi-Newton Trust Region Policy Optimization

Jha, Devesh K.; Raghunathan, Arvind; Romeres, Diego

TR2019-120 October 30, 2019

Abstract

We propose a trust region method for policy optimization that employs Quasi-Newton approximation for the Hessian, called Quasi-Newton Trust Region Policy Optimization (QNTRPO). Gradient descent is the de facto algorithm for reinforcement learning tasks with continuous controls. The algorithm has achieved state-of-the-art performance when used in reinforcement learning across a wide range of tasks. However, the algorithm suffers from a number of drawbacks including: lack of stepsize selection criterion, and slow convergence. We investigate the use of a trust region method using dogleg step and a Quasi-Newton approximation for the Hessian for policy optimization. We demonstrate through numerical experiments over a wide range of challenging continuous control tasks that our particular choice is efcient in terms of number of samples and improves performance.

Conference on Robot Learning (CoRL)

Mitsubishi Electric Research Laboratories, Inc.
201 Broadway, Cambridge, Massachusetts 02139

Quasi-Newton Trust Region Policy Optimization

Devesh K. Jha
MERL
Cambridge, MA
jha@merl.com

Arvind U. Raghunathan
MERL
Cambridge, MA
raghunathan@merl.com

Diego Romeres
MERL
Cambridge, MA
romeres@merl.com

Abstract: We propose a trust region method for policy optimization that employs Quasi-Newton approximation for the Hessian, called Quasi-Newton Trust Region Policy Optimization (QNTRPO). Gradient descent is the de facto algorithm for reinforcement learning tasks with continuous controls. The algorithm has achieved state-of-the-art performance when used in reinforcement learning across a wide range of tasks. However, the algorithm suffers from a number of drawbacks including: lack of stepsize selection criterion, and slow convergence. We investigate the use of a trust region method using dogleg step and a Quasi-Newton approximation for the Hessian for policy optimization. We demonstrate through numerical experiments over a wide range of challenging continuous control tasks that our particular choice is efficient in terms of number of samples and improves performance.

Keywords: Reinforcement Learning, Trust Region Optimization, Quasi-Newton Methods, Policy Gradient

1 Introduction

Reinforcement Learning (RL) is a learning framework that handles sequential decision-making problems, wherein an 'agent' or decision maker learns a policy to optimize a long-term reward by interacting with the (unknown) environment. At each step, an RL agent obtains evaluative feedback (called reward or cost) about the performance of its action, allowing it to improve (maximize or minimize) the performance of subsequent actions [1]. Recent research has resulted in remarkable success of these algorithms in various domains like computer games [2, 3], robotics [4, 5], etc. Policy gradient algorithms can directly optimize the cumulative reward and can be used with a lot of different non-linear function approximators including neural networks. Consequently, policy gradient algorithms are appealing for a lot of different applications, and are widely used for a lot of robotic applications [6, 7, 8]. As a result, it has attracted significant attention in the research community where several new algorithms have been proposed to solve the related problems. However, several problems remain open including monotonic improvement in performance of the policy, selecting the right learning rate (or step-size) during optimization, etc.

Notably, the Trust Region Policy Optimization (TRPO) has been proposed to provide monotonic improvement of policy performance [9]. TRPO relies on a linear model of the objective function and quadratic model of the constraints to determine a candidate search direction. Even though a theoretically justified trust region radius is derived such a radius cannot be computed and hence, linesearch is employed for obtaining a stepsize that ensures progress to a solution. Consequently, TRPO is a scaled gradient descent algorithm and is not a trust region algorithm as the name suggests. More importantly, TRPO does not inherit the flexibility and convergence guarantees provided by the trust region framework [10]. As a consequence, the impact of trust region algorithms have not been fully investigated in the context of policy optimization.

Our objective in this work is to show that a *classical trust region method* in conjunction with *quadratic model* of the objective addresses the drawbacks of TRPO. It is well known that incorporating curvature information of the objective function (i.e. quadratic approximation) allows for rapid convergence in the neighborhood of a solution. Far from a solution, the curvature information should be incorporated in a manner that ensures the search direction improves on the reduction

3rd Conference on Robot Learning (CoRL 2019), Osaka, Japan.

obtained by a linear model. We propose the *Quasi-Newton Trust Region Policy Optimization* (QNTRPO) which uses a dogleg method for computing the step, i.e. both the search direction and stepsize are determined jointly. The Quasi-Newton (QN) method allows for incorporating curvature information by approximating the Hessian of the objective without the need for computing exact second derivatives. In particular, we employ the *classical BFGS* approximation [10]. The dogleg method is well known to produce at least as much reduction obtained using a linear model [10], thus ensuring that QNTRPO does at least as well as the TRPO. The choice of QN method and search direction are chosen to ensure that global convergence properties are retained and the computational cost is comparable to that of TRPO. We want to investigate if QNTRPO, which has a different step from TRPO, can

(i) accelerate the convergence to an optimal policy, and

(ii) achieve better performance in terms of average reward.

QNTRPO computes the stepsize as part of the search direction computation and stepsize is naturally varied according to the accuracy of the quadratic model of the objective. QNTRPO learns faster than TRPO due to the quadratic model and improved search direction. We test the proposed method on several challenging locomotion tasks for simulated robots in the OpenAI Gym environment. We compare the results against the original TRPO algorithm and show that we can consistently achieve better learning rate as well as performance.

2 Background

We first introduce notation and summarize the standard policy gradient framework for RL and the TRPO problem.

2.1 Notation

We address policy learning in continuous/discrete action spaces. We consider an infinite horizon Markov decision process (MDP) defined by the tuple $(\mathcal{S}, \mathcal{A}, P, r, \gamma)$, where the state space $\mathcal{S}$ is continuous, and the unknown state transition probability $P : \mathcal{S} \times \mathcal{S} \times \mathcal{A} \to [0, 1]$ represents the probability density of the next state $s_{t+1} \in \mathcal{S}$ given the current state $s_t \in \mathcal{S}$ and action $a_t \in \mathcal{A}$ and γ is the standard discount factor. The environment emits a reward $r : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$ on each transition.

Let π denote a stochastic policy $\pi : \mathcal{S} \times \mathcal{A} \to [0, 1]$, and let $\eta(\pi)$ denote the expected discounted reward:

$$\eta(\pi) = \mathbb{E}_{s_0, a_0, \ldots}\left[\sum_{t=0}^{\infty} \gamma^t r(s_t)\right], \text{ where } s_0 \sim \rho_0(s_0), a_t \sim \pi(a_t|s_t), s_{t+1} \sim P(s_{t+1}|s_t, a_t).$$

where, ρ_0 is the state distribution of the initial state s_0. Then, we use the standard definition of the state-action value function Q_π, the state value function V_π, advantage function A_π, and the unnormalized discount visitation frequencies ρ_π:

$$Q_\pi(s_t, a_t) = \mathbb{E}_{s_{t+1}, a_{t+1}, \ldots}\left[\sum_{l=0}^{\infty} \gamma^l r(s_{t+l})\right], \quad V_\pi(s_t) = \mathbb{E}_{a_t, s_{t+1}, \ldots}\left[\sum_{l=0}^{\infty} \gamma^l r(s_{t+l})\right].$$

$$A_\pi(s, a) = Q_\pi(s, a) - V_\pi(s), \quad \rho_\pi(s) = \sum_{t=0}^{\infty} \gamma^t Pr(s_t = s|\pi, \rho_0)$$

where in the definition of ρ_π, $s_0 \sim \rho_0$ and the actions are chosen according to π.

In [11], the authors derived an expression for the expected return of the another policy $\tilde{\pi}$ in terms of advantage over π, accumulated over timesteps:

$$\eta(\tilde{\pi}) \quad = \eta(\pi) + \mathbb{E}_{s_0, a_0, \ldots, \sim \tilde{\pi}}\left[\sum_{t=0}^{\infty} \gamma^t A_\pi(s_t, a_t)\right] = \eta(\pi) + \sum_s \rho_{\tilde{\pi}}(s) \sum_a \tilde{\pi}(a|s) A_\pi(s, a). \quad (1)$$

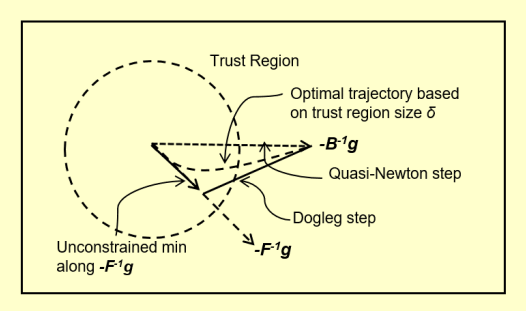


Figure 1: Exact and Dogleg approximation for Trust Region Optimization

A local approximation to $\eta(\tilde{\pi})$ can then be obtained by making an approximation of the state-visitation frequency using the policy π which is expressed as

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

In [9], the authors present an algorithm to maximize $L_\pi(\tilde{\pi})$ using a constrained optimization approach. For simplicity, we denote $L_\pi(\tilde{\pi})$ as $L_{\theta_{\text{old}}}(\theta)$, where θ represents the policy parameters.

2.2 Trust Region Policy Optimization (TRPO)

In this section, we first describe the original TRPO problem and then we present our proposed method to contrast the difference in the optimization techniques. Using several simplifications to the conservative iteration proposed in [11], authors in [9] proposed a practical algorithm for solving the policy gradient problem using generalized advantage estimation [12]. In the TRPO, the following constrained problem is solved at every iteration:

$$\text{maximize } L_{\theta_{\text{old}}}(\theta) \text{ subject to } \bar{D}_{KL}(\theta_{\text{old}}, \theta) \leq \delta$$

where $L_{\theta_{\text{old}}}(\theta)$ is the following term.

$$L_{\theta_{\text{old}}}(\theta) = \sum_s \rho_{\theta_{\text{old}}}(s) \sum_a \pi_\theta(a|s) A_{\pi_{\theta_{\text{old}}}}(s,a)$$

For simplicity of notation, we will denote $L_{\theta_{\text{old}}}(\theta)$ as $L(\theta)$ in the following text. The optimization algorithm in TRPO works in two steps: (1) compute a search direction, using a linear model of the objective and quadratic model to the constraint; and (2) perform a line search in that direction, ensuring that we improve the nonlinear objective while satisfying the nonlinear constraint. The search direction in TRPO and its variants is $\Delta\theta = \alpha F^{-1} g$ where $g = \nabla L(\theta)$ is gradient of $L(\theta)$ evaluated at θ_{old} and F is the Fisher information matrix, i.e., the quadratic model to the KL divergence constraint $\bar{D}_{KL}(\theta_{\text{old}}, \theta) = \frac{1}{2}(\theta - \theta_{\text{old}})^T F (\theta - \theta_{\text{old}})$ and F is the Hessian of the KL divergence estimation evaluated at θ_{old}.

In contrast, the proposed algorithm approximates the objective by a quadratic model and uses the Dogleg method [10] to compute a step. Figure 1 depicts the idea behind the Dogleg approximation for the trust region optimum. As seen in Figure 1 the Dogleg method smoothly transitions between the scaled gradient step and a Quasi-Newton step, which is the unconstrained minimizer of the quadratic model. Thus, the step automatically changes direction depending on the size of the trust region. The size of the trust region is modified according to the accuracy of the quadratic model to ensure global convergence of the algorithm.

3 Quasi-Newton Trust Region Method (QNTRM)

QNTRM has three distinctive elements that sets it apart from TRPO. First, the use of a quadratic approximation for the objective via a Quasi-Newton approximation of the Hessian. Second, the Dogleg method that defines the step. Finally, the adaptive change of the stepsize through the classical trust region framework. We describe each of these in the following. In the rest of the paper, let

$f(\theta) = -L(\theta)$ so that maximization of $L(\theta)$ can be equivalently expressed as minimization of $f(\theta)$. We use θ_k to refer to the value of the parameters at the k-th iterate of the algorithm. For sake of brevity, f_k denotes $f(\theta_k)$, ∇f_k denotes $\nabla f(\theta_k)$ and $\nabla^2 f_k$ denotes $\nabla^2 f(\theta_k)$.

3.1 Quadratic Approximation via BFGS

QNTRM approximates the objective using a quadratic model $f_k^q(\theta)$ defined as

$$f_k^q(\theta) = f_k + \nabla f_k^T(\theta - \theta_k) + \frac{1}{2}(\theta - \theta_k)^T B_k(\theta - \theta_k)$$

where $B_k \approx \nabla^2 f_k$ is an approximation to the Hessian of f at the point θ_k. We employ the BFGS approximation [10] to obtain B_k. Starting with an initial symmetric positive definite matrix B_0, the approximation B_{k+1} for $k \geq 0$ is updated at each iteration of the algorithm using the step s_k and a difference of the gradients of f along the step $y_k = \nabla f(\theta_k + s_k) - \nabla f_k$. The update B_{k+1} is the smallest update (in Frobenius norm $\|B - B_k\|_F$) to B_k such that $B_{k+1}s_k = y_k$ (i.e. the secant condition holds), and B_{k+1} is symmetric positive definite, i.e.

$$B_{k+1} = \arg\min_B \|B - B_k\|_F \text{ subject to } Bs_k = y_k,\ B = B^T.$$

The above minimization can be solved analytically [10] and the update step is

$$B_{k+1} = B_k - \frac{B_k s_k s_k^T B_k}{s_k^T B_k s_k} + \frac{y_k y_k^T}{y_k^T s_k} \tag{2}$$

Observe the effort involved in performing the update is quite minimal. The above update does not enforce positive definiteness of B_{k+1}. By recasting (2) after some algebraic manipulation as

$$B_{k+1} = \left(I - \frac{1}{s_k^T B_k s_k} B_k s_k s_k^T\right) B_k \left(I - \frac{1}{s_k^T B_k s_k} s_k s_k^T B_k\right) + \frac{y_k y_k^T}{y_k^T s_k}$$

it is easy to see that B_{k+1} is positive definite as long as $y_k^T s_k > 0$.

3.2 Dogleg Method

The search direction in QNTRM $\Delta\theta_k$ is computed by approximately solving

$$\min_{\Delta\theta} f_k^q(\theta_k + \Delta\theta) \text{ subject to } (\Delta\theta)^T F_k(\Delta\theta) \leq \delta_k$$

i.e. minimizing the quadratic model of the objective subject to the KL-divergence constraint. The above problem is only solved approximately since the goal is only to produce a search direction $\Delta\theta_k$ that furthers the overall objective of minimizing $f(\theta)$ at moderate computational cost. However, the search direction $\Delta\theta_k$ should incorporate both the curvature and attain sufficient progress towards a solution. In fact, we desire at least as much progress as the step in TRPO. The Dogleg method does precisely this by combining the scaled gradient direction $\Delta\theta_k^{GD} = -\beta_k F_k^{-1}\nabla f_k$ and the QN direction $\Delta\theta_k^{QN} = -B_k^{-1}\nabla f_k$. The search direction $\Delta\theta_k^{DL}$ is obtained using Algorithm 1.

The algorithm first computes the QN direction $\Delta\theta_k^{QN}$ and accepts it if the trust region constraint defined by the KL-divergence holds (Step 4). If not the algorithm computes the scaled gradient direction (Step 5) and a stepsize β_k so as to minimize the quadratic model, i.e.

$$\beta_k = \frac{\nabla f_k^T F_k^{-1}\nabla f_k}{(F_k^{-1}\nabla f_k)^T B_k (F_k^{-1}\nabla f_k)}. \tag{3}$$

Unlike the TRPO, observe that due to the curvature in the objective we can now define an *optimal stepsize* for the gradient direction. If the gradient direction scaled by the optimal stepsize exceeds the trust region then it is further scaled back until the trust region constraint is satisfied and accepted (Step 7). If neither of the above hold then the direction is obtained as a convex combination of the two directions $\Delta\theta(\tau_k) := (\Delta\theta_k^{GD} + \tau_k(\Delta\theta_k^{QN} - \theta_k^{GD}))$. This is the *Dogleg direction*. The parameter τ_k is chosen so that the direction $\Delta\theta(\tau_k)$ satisfies the trust region constraint as an equality (Step 10). The computation of τ_k requires finding the roots of a quadratic equation which can be obtained easily.

Note that QNTRM requires the solution of linear system in order to compute $B_k^{-1}\nabla f_k$ and $F_k^{-1}\nabla f_k$. Both of these can be accomplished by the Conjugate Gradient (CG) method since B_k, F_k are both positive definite. Thus, the computation QNTRM differs from TRPO by an extra CG solve and hence, comparable in computational complexity.

Algorithm 1: Dogleg Method

Data: ∇f_k, B_k, F_k, δ_k
Result: Dogleg direction $\Delta\theta_k^{DL}$

Compute QN direction $\Delta\theta_k^{QN} = -B_k^{-1}\nabla f_k$;
if $(\Delta\theta_k^{QN})^T F_k(\Delta\theta_k^{QN}) \leq \delta_k$ **then**
  return $\Delta\theta_k^{QN}$
end
Compute Gradient direction $\Delta\theta_k^{GD} = -\beta_k F_k^{-1}\nabla f_k$ where β_k is defined in (3);
if $(\Delta\theta_k^{GD})^T F_k(\Delta\theta_k^{GD}) \geq \delta_k$ **then**
  return $\sqrt{\frac{\delta_k}{(\Delta\theta_k^{GD})^T F_k(\Delta\theta_k^{GD})}}\Delta\theta_k^{GD}$
end
Find largest $\tau_k \in [0, 1]$ such that $\Delta\theta(\tau_k) := (\Delta\theta_k^{GD} + \tau_k(\Delta\theta_k^{QN} - \theta_k^{GD}))$ satisfies $(\Delta\theta(\tau_k))^T F_k(\Delta\theta(\tau_k)) = \delta_k$;
return $(\Delta\theta_k^{GD} + \tau_k(\Delta\theta_k^{QN} - \theta_k^{GD}))$;

3.3 Trust Region Algorithm

QNTRM combines the curvature information from QN approximation and Dogleg step within the framework of the classical trust region algorithm. The algorithm is provided in Algorithm 2 and incorporates safeguards to ensure that B_k's are all positive definite. At each iteration of the algorithm, a step $\Delta\theta_k^{DL}$ is computed using Algorithm 1 (Step 3). The trust region algorithm accepts or rejects the step based on a measure of how well the quadratic model approximates the function f along the step $\Delta\theta_k^{DL}$. The commonly used measure [10] is the ratio of the actual decrease in the objective and the decrease that is predicted by the quadratic model (Step 4). If this ratio ν_k is close to or larger than 1 then the step computed using the quadratic model provides a decrease in f that is comparable or much better than predicted by the model. The algorithm uses this as an indication that the quadratic model approximates f well. Accordingly, if the ratio (Step 4) is larger than a threshold ($\underline{\nu}$), the parameters are updated (Step 6). If in addition, the ratio is larger than $\overline{\nu}$ and $\Delta\theta_k$ satisfies the trust region size as an equality then the size of the trust region is increased in the next iteration (Step 8). This condition indicates that the quadratic model matches the objective f with high accuracy and that the progress is being impeded by the size of the trust region. Hence, the algorithm increases the trust region for the next iteration. With the increased trust region size the algorithm promotes the possible acceptance of a direction other than the scaled gradient direction. On the other hand, if the ratio is below $\underline{\nu}$ then the computed direction is rejected (Step 11) and the size of the trust region is decreased (Step 12). This reflects the situation that the quadratic model does not the capture the objective variation well. Note that as the size of the trust region decreases the performance of the algorithm mirrors that of TRPO very closely. Thus, QNTRM is naturally designed to be no worse than the TRPO and often surpass TRPO's performance whenever the quadratic model approximates the objective function well. Finally, we update the QN approximation whenever the $s_k^T y_k$ is greater than a minimum threshold. This ensures that the matrices B_k are all positive definite (Step 16). Note that this safeguard is necessary since the Dogleg step cannot be designed to ensure that $s_k^T y_k > 0$.

4 Quasi-Newton Trust Region Policy Optimization (QNTRPO)

QNTRPO is the trust region algorithm that we propose in this paper for policy optimization, The algorithm differs from TRPO in the step that is computed at every iteration of policy iteration. For completeness of the paper, it is presented as an Algorithm 3. It is noted that the only difference between QNTRPO and TRPO is the way the trust region optimization problem is solved (see line 4 in Algorithm 3). It is noted that in the original TRPO formulation, the line 4 in Algorithm 3 is performed using the scaled gradient method as discussed earlier. This is the major difference between the proposed and the algorithm proposed in TRPO. Note that QNTRM is an iterative procedure and that the step for every iteration of Algorithm 3 is computed by iterating over K steps of QNTRM (see Algorithm 2). This is yet another difference over TRPO where a single gradient descent step is computed for each episode. As a result, the computational time per episode for QNTRPO is no

Algorithm 2: Quasi-Newton Trust Region Method (QNTRM)

Data: Parameters of algorithm – $0 < \underline{\nu} < \overline{\nu} < 1$, $\overline{\delta} \in (0,1)$, $\underline{\kappa} \in (0,1)$, $0 < \underline{\omega} < 1 < \overline{\omega}$.
Data: Initial policy parameters – θ_0
Data: Convergence tolerance – $\epsilon > 0$, Limit on iterations K
Result: θ^*

Set $k = 0$;
while $\|\nabla f_k\| > \epsilon$ *and* $k < K$ **do**
  Compute the Dogleg step $\Delta\theta_k^{DL}$ using Algorithm 1;
  Compute $\nu_k = \frac{f(\theta_k + \Delta\theta_k^{DL}) - f(\theta_k)}{f_k^q(\theta_k + \Delta\theta_k^{DL}) - f_k^q(\theta_k)}$;
  if $\nu_k \geq \underline{\nu}$ **then**
    Set $\theta_{k+1} = \theta_k + \Delta\theta_k^{DL}$;
    if $\nu_k \geq \overline{\nu}$ *and* $(\Delta\theta_k^{DL})^T F_k(\Delta\theta_k^{DL}) = \delta_k$ **then**
      Set $\delta_{k+1} = \min(\overline{\delta}, \overline{\omega} \cdot \delta_k)$;
    end
  else
    Set $\theta_{k+1} = \theta_k$;
    Set $\delta_{k+1} = \underline{\omega} \cdot \delta_k$;
  end
  Set $s_k = \Delta\theta_k^{DL}$ and $y_k = \nabla f(\theta_k + \Delta\theta_k^{DL}) - \nabla f(\theta_k)$;
  if $s_k^T y_k \geq \underline{\kappa}$ **then**
    Update B_{k+1} using (2);
  else
    Set $B_{k+1} = B_k$;
  end
  Set $k = k + 1$;
end
return $\theta^* = \theta_k$

more than $(2 \times K)$ that of TRPO owing to the possibly two linear systems solves in Dogleg method and K iterations in QNTRM.

Algorithm 3: QNTRPO

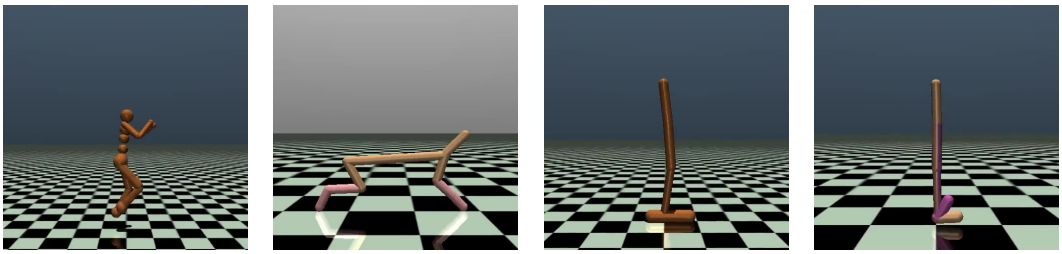
Initialize policy parameters θ^0
for $i = 0, 1, 2, \ldots$ *until convergence* **do**
  Compute all Advantage values $A_{\pi_{\theta^i}}(s, a)$ and state-visitation frequency ρ_{θ^i};
  Define the objective function for the episode $L_{\theta^i}(\theta) = -f^i(\theta)$;
  Obtain θ^{i+1} using QNTRM to minimize $f^i(\theta)$ with initial policy parameters $\theta_0 = \theta^i$
end
;

5 Experimental Results

In this section, we present experimental results for policy optimization using several different environments for continuous control from the openAI Gym benchmark [13]. In these experiments, we try to answer the following questions:

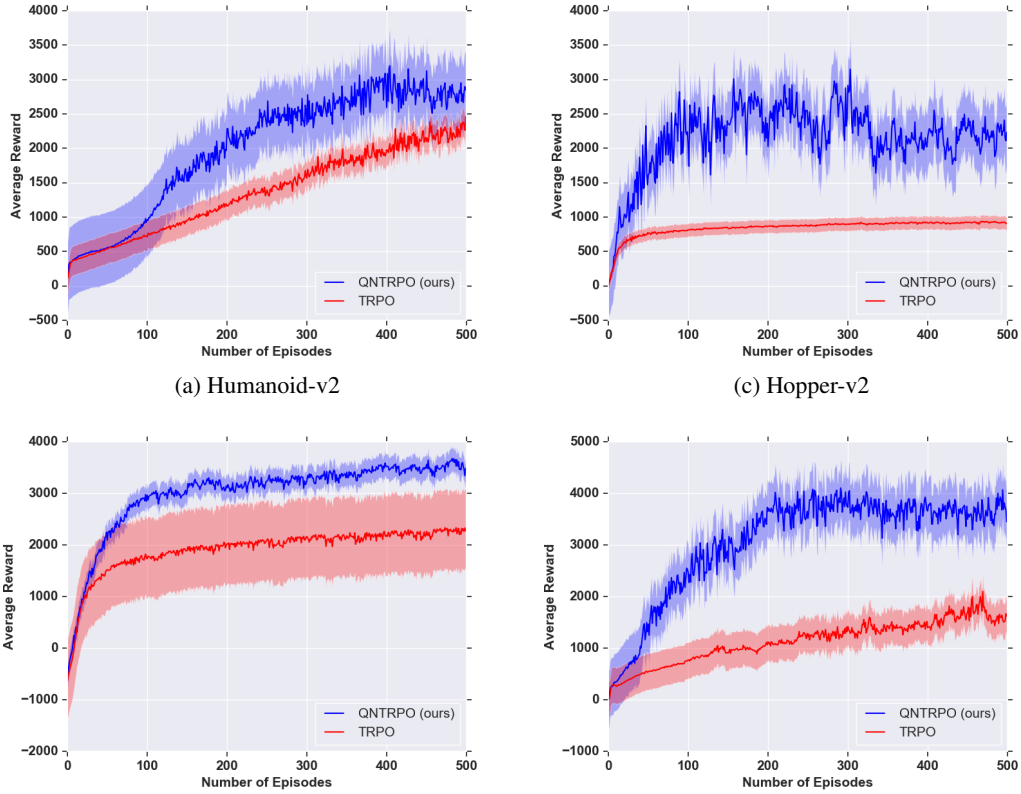
1. Can QNTRPO achieve better learning rate (sample efficiency) than TRPO consistently over a range of tasks?
2. Can QNTRPO achieve better performance than TRPO over a range of tasks in terms of average reward?

In the following, we try to answer these two questions by evaluating our algorithm on several continuous control tasks. In particular, we investigate and present results on four different environments



(a) Humanoid-v2 (b) HalfCheetah-v2 (c) Hopper-v2 (d) Walker 2d-v2

Figure 2: The four continuous control benchmark tasks considered in this paper.



(a) Humanoid-v2 (c) Hopper-v2

(b) HalfCheetah-v2 (d) Walker 2d-v2

Figure 3: Results of our method compared against the TRPO method in [9] compared on four benchmark continuous control environments in OpenAI gym. The plots show the average batch reward obtained by both methods averaged over three different runs.

in Mujoco physics simulator [14]. We implement four locomotion tasks of varying dynamics and difficulty: Humanoid [15, 5], Half-Cheetah [16], Walker [17] and Hopper [9]. The goal for all these tasks is to move forward as quickly as possible. These tasks have been proven to be challenging to learn due to the high degrees of freedom of the robots [5]. A great amount of exploration is needed to learn to move forward without getting stuck at local minima. During the initial learning stages, it is easy for the algorithm to get stuck in a local minima as the controls are penalized and the robots have to avoid falling. The state and action dimensions of these tasks are listed in Table 1.

We run both TRPO and QNTRPO for 500 episodes and average all results across five different runs with different random seeds for the environment initialization. All hyperparameters for the algorithms – batch size, policy network architecture, step size and the generalized advantage estimation

	Humanoid-v2	HalfCheetah-v2	Walker2d-v2	Hopper-v2
State Dimension	376	17	17	11
Action Dimension	17	6	6	3

Table 1: State and action dimensions of the RL tasks considered in the paper.

coefficient (λ) – are identical for both algorithms. As TRPO (and thus QNTRPO) performs better with bigger batches, we use a batch size of 15000. In each of these episodes, trajectories are generated for a maximum length of 2000 and then restarted either if the terminal condition is met or the trajectory length is satisfied. The network architecture is kept the same across all the tasks. The trust region radius is chosen to be 0.1 (note that this is the parameter $\bar{\delta}$ in Algorithm 2). At lower trust region radius both algorithms performed slower and thus the results are not reported here. The discount factor γ is chosen to be 0.99 and the constant λ for advantage function estimation is chosen to be 0.97. The parameters for QNTRM were chosen to be the following: $K = 10$, $\overline{\nu} = 0.75$, $\underline{\nu} = 0.1$, $\underline{\omega} = 0.3$, $\overline{\omega} = 2$ and $\underline{\kappa} = 10^{-3}$. Codes for running these experiments are available at www.merl.com/research/license#QNTRPO.

Results of our experiments are shown in Figure 3. For all four tasks, we can demonstrate that QNTRPO can achieve faster learning, and thus better sample efficiency than the original TRPO. Furthermore, the performance of QNTRPO is also significantly better than TRPO. This is evident from the fact that QNTRPO achieves higher rewards than TRPO, which also has longer transitory. For high complexity problems like Humanoid, QNTRPO takes about 350 episodes with the current batch size to reach the maximum score (of around 3000). These results show that QNTRPO can calculate a better step for the constrained optimization problem for policy iteration using QNTRM.

6 Conclusions and Future Work

In this paper, we presented an algorithm for policy iteration using a Quasi-Newton trust region method. The problem was inspired by the policy optimization problem formulated in [9] where a linesearch is performed to compute the step size in the direction of steepest descent using a quadratic model of the constraint. In this paper, we proposed a dogleg method for computing the step during policy iteration which has theoretical guarantees [10] of better performance over the scaled gradient descent method used in [9]. The proposed method was compared against the original TRPO algorithm in four different continuous control tasks in Mujoco physics simulator. The proposed algorithm outperformed TRPO in learning speed as well performance indicating that the proposed method can compute better step for the policy optimization problem.

Despite the good performance , there are a number of open issues for which we do not have a complete understanding. We have observed that the maximum trust region radius ($\bar{\delta}$) plays an important role in speed of learning. However, choosing this arbitrarily might result in poor convergence. Furthermore, to achieve monotonic improvement in policy performance, one has to select the trust region radius very carefully which is undesirable. It would also be interesting to study the interplay of batch size and trust region radius. This can help address the issue of steplength selection. In the future, we would like to further investigate several features of the proposed algorithm including the following.

- Analyze the stability of the proposed algorithm to the size of trust region radius and batch size. We believe that the proposed method can be used to fine tune the hyperparameter of trust region radius which controls the maximum step size in each iteration of the algorithm.
- Evaluate the proposed algorithm on much higher dimension learning problem for end-to-end learning using a limited memory version of the proposed algorithm.
- Use ideas from ensemble methods [18], scalable bootstrapping [19] and factored methods to curvature [20] for better and efficient approximation of the objective function.
- Evaluate the performance on challenging, sparse reward environments [21, 22].

References

[1] R. S. Sutton and A. G. Barto. *Reinforcement learning: An introduction (2nd Edition)*, volume 1. MIT press Cambridge, 2018.

[2] D. Silver, A. Huang, C. J. Maddison, A. Guez, L. Sifre, G. Van Den Driessche, J. Schrittwieser, I. Antonoglou, V. Panneershelvam, M. Lanctot, et al. Mastering the game of go with deep neural networks and tree search. *nature*, 529(7587):484, 2016.

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

[4] S. Levine, C. Finn, T. Darrell, and P. Abbeel. End-to-end training of deep visuomotor policies. *The Journal of Machine Learning Research*, 17(1):1334–1373, 2016.

[5] Y. Duan, X. Chen, R. Houthooft, J. Schulman, and P. Abbeel. Benchmarking deep reinforcement learning for continuous control. In *International Conference on Machine Learning*, pages 1329–1338, 2016.

[6] J. Peters and S. Schaal. Policy gradient methods for robotics. In *2006 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 2219–2225. IEEE, 2006.

[7] J. Kober and J. R. Peters. Policy search for motor primitives in robotics. In *Advances in neural information processing systems*, pages 849–856, 2009.

[8] M. P. Deisenroth, G. Neumann, J. Peters, et al. A survey on policy search for robotics. *Foundations and Trends® in Robotics*, 2(1–2):1–142, 2013.

[9] J. Schulman, S. Levine, P. Abbeel, M. Jordan, and P. Moritz. Trust region policy optimization. In *International Conference on Machine Learning*, pages 1889–1897, 2015.

[10] J. Nocedal and S. J. Wright. *Numerical Optimization*. Springer, second edition, 2006.

[11] S. Kakade and J. Langford. Approximately optimal approximate reinforcement learning. 2002.

[12] J. Schulman, P. Moritz, S. Levine, M. Jordan, and P. Abbeel. High-dimensional continuous control using generalized advantage estimation. *arXiv preprint arXiv:1506.02438*, 2015.

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

[14] E. Todorov, T. Erez, and Y. Tassa. Mujoco: A physics engine for model-based control. In *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 5026–5033. IEEE, 2012.

[15] Y. Tassa, T. Erez, and E. Todorov. Synthesis and stabilization of complex behaviors through online trajectory optimization. In *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 4906–4913. IEEE, 2012.

[16] N. Heess, G. Wayne, D. Silver, T. Lillicrap, T. Erez, and Y. Tassa. Learning continuous control policies by stochastic value gradients. In *Advances in Neural Information Processing Systems*, pages 2944–2952, 2015.

[17] S. Levine and V. Koltun. Guided policy search. In *International Conference on Machine Learning*, pages 1–9, 2013.

[18] A. Rajeswaran, S. Ghotra, B. Ravindran, and S. Levine. Epopt: Learning robust neural network policies using model ensembles. *arXiv preprint arXiv:1610.01283*, 2016.

[19] A. Kleiner, A. Talwalkar, P. Sarkar, and M. I. Jordan. A scalable bootstrap for massive data. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 76(4):795–816, 2014.

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

[21] J. v. Baar, A. Sullivan, R. Cordorel, D. Jha, D. Romeres, and D. Nikovski. Sim-to-real transfer learning using robustified controllers in robotic tasks involving complex dynamics. In *2019 International Conference on Robotics and Automation (ICRA)*, pages 6001–6007, May 2019. doi:10.1109/ICRA.2019.8793561.

[22] D. Romeres, D. K. Jha, A. DallaLibera, B. Yerazunis, and D. Nikovski. Semiparametrical gaussian processes learning of forward dynamical models for navigating in a circular maze. In *2019 International Conference on Robotics and Automation (ICRA)*, pages 3195–3202, May 2019. doi:10.1109/ICRA.2019.8794229.

A Derivation of the Dogleg step for QNTRM

The Dogleg method aims to obtain an approximate solution of the trust region problem

$$\min_{\Delta\theta} f_k^q(\theta_k + \Delta\theta) \text{ subject to } (\Delta\theta)^T F_k(\Delta\theta) \leq \delta_k \tag{4}$$

where $f_k^q(\theta_k + \Delta\theta) = f_k + \nabla f_k^T(\Delta\theta) + \frac{1}{2}(\Delta\theta)^T B_k(\Delta\theta)$. In this section, we derive the Dogleg step under the trust region defined by the KL-divergence constraint.

We begin by first transforming the trust region problem in (4) into standard form. Let $F_k = L_k L_k^T$ which can be obtained for example by Cholesky factorization since the Fischer matrix F_k is positive definite. Note that the factorization is only used for deriving the step and is never required for the computations.

Defining $\widehat{\Delta\theta} = L_k^T \Delta\theta$ we can recast the quadratic model as

$$\widehat{f}_k^q(\theta_k + \hat{\delta\theta}) = f_k + \widehat{\nabla f}_k^T(\widehat{\Delta\theta}) + \frac{1}{2}(\widehat{\Delta\theta})^T \widehat{B}_k(\widehat{\Delta\theta}) \tag{5}$$

where $\widehat{\nabla f}_k = L_k^{-1}\nabla f_k$ and $\widehat{B}_k = L_k^{-1} B_k L_k^{-T}$. It is easily verified that $f_k^q(\theta_k + \Delta\theta) = \widehat{f}_k^q(\theta_k + L_k^T\Delta\theta)$ and $(\Delta\theta)^T F_k(\Delta\theta) = (\widehat{\Delta\theta})^T(\widehat{\Delta\theta})$. Hence, the trust region problem in (4) can be recast as the standard trust region problem

$$\min_{\widehat{\Delta\theta}} \widehat{f}_k^q(\theta_k + \widehat{\Delta\theta}) \text{ subject to } (\widehat{\Delta\theta})^T(\widehat{\Delta\theta}) \leq \delta_k \tag{6}$$

In the following, we will derive the Quasi-Newton, Gradient and Dogleg steps based on (6) and then, transform these steps to the original space using the transformation $\widehat{\Delta\theta} = L_k^T\Delta\theta$.

The Quasi-Newton step for (6) is

$$\widehat{\Delta\theta}^{QN} = -\widehat{B}_k^{-1}\widehat{\nabla f}_k = -L_k^T B_k^{-1}\nabla f_k \tag{7}$$

where the second equality is obtained by substitution. Thus, the Quasi-Newton step in the original space of parameters is

$$\Delta\theta^{QN} = -B_k^{-1}\nabla f_k. \tag{8}$$

The gradient direction for (6) is $\widehat{\Delta\theta}^{gd} = -\widehat{\nabla f}_k = -L_k^{-1}\nabla f_k$. The optimum stepsize β_k along the gradient direction is obtained from

$$\min_{\beta} \widehat{f}_k^q(\theta_k + \beta\widehat{\Delta\theta}^{gd}). \tag{9}$$

Hence, the optimal stepsize along the gradient direction is

$$\beta_k = \frac{\widehat{\nabla f}_k^T\widehat{\nabla f}_k}{\widehat{\nabla f}_k^T\widehat{B}_k\widehat{\nabla f}_k} = \frac{\nabla f_k^T F_k^{-1}\nabla f_k}{(F_k^{-1}\nabla f_k)^T B_k (F_k^{-1}\nabla f_k)} \tag{10}$$

and the scaled gradient direction is

$$\widehat{\Delta\theta}^{GD} = -\beta_k \widehat{\nabla f}_k. \tag{11}$$

Thus, the scaled gradient step in the original space of parameters is

$$\Delta\theta^{GD} = -\beta_k F_k^{-1} \nabla f_k. \tag{12}$$

The Dogleg step for (6) computes a τ_k such that

$$\begin{aligned} & \left\| \widehat{\Delta\theta}^{GD} + \tau_k (\widehat{\Delta\theta}^{QN} - \widehat{\Delta\theta}^{GD}) \right\|^2 && = \delta_k \\ \implies & \left\| L_k^T \Delta\theta^{GD} + \tau_k (L_k^T \Delta\theta^{QN} - L_k^T \Delta\theta^{GD}) \right\|^2 && = \delta_k \\ \implies & \Delta\theta(\tau_k) F_k \Delta\theta(\tau_k) && = \delta_k \end{aligned} \tag{13}$$

where $\Delta\theta(\tau_k) = \Delta\theta^{GD} + \tau_k(\Delta\theta^{QN} - \Delta\theta^{GD})$.

B Time performance comparison

In Table 2 we compare the wall clock time for each of the four tasks. For each task we average the time needed to perform each single episode over all the episodes. The performance are computed on a Linux desktop with i7-6700K Intel Core.

Algorithm	Humanoid-v2	HalfCheetah-v2	Hopper-v2	Walker2d-v2
TRPO	9.68 ± 0.13 $[s]$	3.19 ± 0.018 $[s]$	3.79 ± 0.04 $[s]$	4.29 ± 0.06 $[s]$
QNTRPO	91.99 ± 10.87 $[s]$	54.66 ± 8.65 $[s]$	30.02 ± 7.22 $[s]$	37.98 ± 6.78 $[s]$

Table 2: Average and standard deviation in seconds of wall clock time for each episode of all the experiments for the 4 environments on a Linux desktop with i7-6700K Intel Core.

QNTRPO is slower than the standard TRPO due to multiple inner iteratins that are performed for each episode. The time performance is consistent with the computational analysis described in the paper.

The QNTRM is an iterative procedure and the step for every iteration of Algorithm 3 is computed by iterating over K steps of QNTRM (see Algorithm 2). Instead, in TRPO a single gradient descent step is computed for each episode. As a result, the computational time per episode for QNTRPO is no more than $(2 \times K)$ that of TRPO owing to the possibly two linear systems solves in Dogleg method and K iterations in QNTRM. In our experiments K is chosen to be 10 and it is clear from Table 2 that the ratio in performance time between QNTRPO and TRPO is below 20.