Supplementary Material for 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

1 Derivation of the Dogleg step for QNTRM

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

$$\min_{\Delta\theta} f_k^q(\theta_k + \Delta\theta) \text{ subject to } (\Delta\theta)^T F_k(\Delta\theta) \leq \delta_k \tag{1}$$

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 (1) 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

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

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 (1) can be recast as the standard trust region problem

$$\min_{\widehat{\Delta\theta}} \widehat{f}_k^q(\theta_k + \widehat{\Delta\theta}) \text{ subject to } (\widehat{\Delta\theta})^T(\widehat{\Delta\theta}) \leq \delta_k \tag{3}$$

In the following, we will derive the Quasi-Newton, Gradient and Dogleg steps based on (3) 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 (3) is

$$\widehat{\Delta\theta}^{QN} = -\widehat{B}_k^{-1}\widehat{\nabla f}_k = -L_k^T B_k^{-1}\nabla f_k \tag{4}$$

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

$$\Delta\theta^{QN} = -B_k^{-1}\nabla f_k. \tag{5}$$

The gradient direction for (3) 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

$$\min_{\beta} \widehat{f}_k^q(\theta_k + \beta\widehat{\Delta\theta}^{gd}). \tag{6}$$

Hence, the optimal stepsize along the gradient direction is

$$\beta_k = \frac{\widehat{\nabla f}_k^T\widehat{\nabla f}_k}{\widehat{\nabla f}_k^T\widehat{B}_k\widehat{\nabla f}_k} = \frac{\nabla f_k^T F_k^{-1}\nabla f_k}{(F_k^{-1}\nabla f_k)^T B_k (F_k^{-1}\nabla f_k)} \tag{7}$$

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

and the scaled gradient direction is

$$\widehat{\Delta\theta}^{GD} = -\beta_k \widehat{\nabla f}_k. \tag{8}$$

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

$$\Delta\theta^{GD} = -\beta_k F_k^{-1} \nabla f_k. \tag{9}$$

The Dogleg step for (3) computes a τ_k such that

$$\begin{aligned} & \left\| \widehat{\Delta\theta}^{GD} + \tau_k (\widehat{\Delta\theta}^{QN} - \widehat{\Delta\theta}^{GD}) \right\|^2 && = \delta_k \\ \implies & \left\| L_k^T \Delta\theta^{GD} + \tau_k (L_k^T \Delta\theta^{QN} - L_k^T \Delta\theta^{GD}) \right\|^2 && = \delta_k \\ \implies & \Delta\theta(\tau_k) F_k \Delta\theta(\tau_k) && = \delta_k \end{aligned} \tag{10}$$

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

2 Time performance comparison

In Table 1 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 1: 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 1 that the ratio in performance time between QNTRPO and TRPO is below 20.