1. Very Basic of Numerical Optimization for TRPO

# number	1
Select	appendix

Introduction

In Trust-Region Policy Optimization(TRPO), numerical optimization algorithms(techniques) like line searching, trust region method, and conjugate gradient are leveraged to update the parameters efficiently. This post is written to introduce those algorithms briefly to **help** understand TRPO. If you are familiar with

Line Search, Trust-Region Method, Conjugate Gradient,

you may skip this post.

---

1. Line Search Methods

1.1 Outline of the Algorithm

Let's say we want to find the minimum of the function f. Beginning at x_0, optimization algorithms generate a sequence of iterates $\{x_k\}_{k=0}^{\infty}$ that terminate when either no more progress can be made or when it seems that a solution point has been approximated with sufficient accuracy.

ITo determine how to move from x_k to x_{k+1}, the algorithm uses information about f at x_k and possibly from earlier iterates. They use this information to find a new iterate x_{k+1} with a lower function value than x_k.

In the line search approach, the algorithm chooses a ***search direction*** p_k at x_k and ***searches along*** that direction for a new iterate with a lower function value. The *distance* to move along p_k can be found by approximately solving the following one-dimensional minimization problem to find a ***step length*** α ($x_{k+1} = x_k + \alpha p_k$):

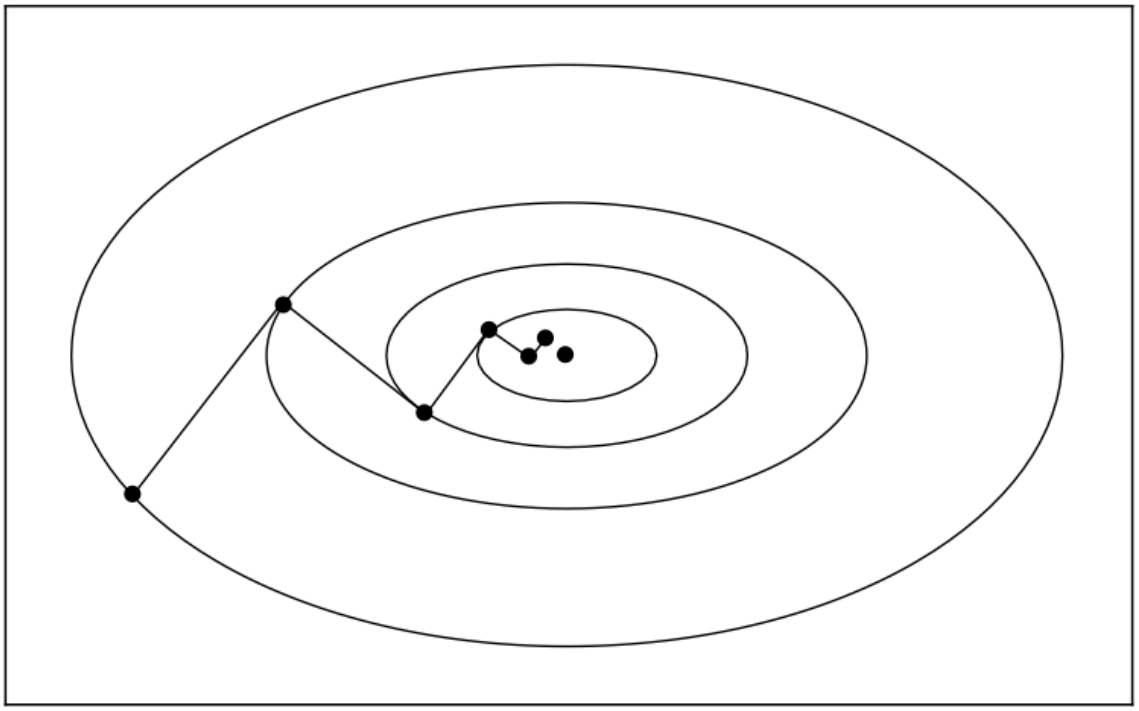
$$\min_{\alpha>0} f(x_k + \alpha p_k).$$

Most line search algorithms require p_k to be a *descent direction*. The search direction often has the form

$$p_k = -B_k^{-1} \nabla f_k$$

where B_k is a symmetric and nonsingular matrix. This matrix depends on the algorithm.

In the steepest descent method B_k is simply the identity matrix I, while in Newton's method B_k is the exact Hessian $\nabla^2 f(x_k)$.



Steepest descent steps

1.2 Step Length Condition

In computing the step length α_k, we face a trade-off. We would like to choose α_k to give a substantial reduction of f, but at the same time, we do not want to spend too much time making the choice. The ideal choice would be the *global minimizer* of the univariate function $\phi(\alpha)$ defined by

$$\phi(\alpha) = f(x_k + \alpha p_k), \alpha > 0$$

but in general, it is too expensive to identify this value. More practical strategies perform an *inexact* line search to identify a step length that achieves adequate reductions in f at minimal cost.

The Wolfe Conditions

The Wolfe Conditions, which is a popular inexact line search condition stipulates that α_k should first of all give a *sufficient decrease* in the objective function f , as measured by the following inequality:

$$f(x_k + \alpha p_k) \leq f(x_k) + c_1 \alpha \nabla f_k^T p_k$$

for some constant $c_1 \in (0, 1)$. In practice, c_1 is chosen to be quite small, say $c_1 = 10^{-4}$.

For the curvature condition, it is required that α_k to satisfy

$$\nabla f(x_k + \alpha_k p_k)^T p_k \geq c_2 \nabla f_k^T p_k$$

for $0 < c_1 < c_2 < 1$. Typical values of c_2 are 0.9 when the search direction is chosen by Newton or quasi-Newton method, and 0.1 when the search direction is obtained from a nonlinear conjugate gradient method.

1.3 Search Direction

Let the angle θ_k between p_k and the steepest descent direction $-\nabla f_k$ defined by

$$\cos \theta_k = \frac{-\nabla f_k^T p_k}{\|\nabla f_k\| \|p_k\|}.$$

Zoutendijk's Condition

Consider any iteration of the form $x_{k+1} = x_k + \alpha p_k$, where p_k is a descent direction and α_k satisfies the Wofle Conditions. Suppose that f is bounded below in R^n and that f is continuously differentiable in an open set N containing the level set $\mathscr{L} := \{x | f(x) \leq f(x_0)\}$, where x_0 is the starting point of the iteration. Assume also that the gradient ∇f is Lipschitz continuous on N, that is, there exists a constant $L > 0$ such that

$$\|\nabla f(x) - \nabla f(\tilde{x})\| \leq L\|x - \tilde{x}\|$$

for all $x, \tilde{x} \in N$.

Then

$$\sum_{k \geq 0} \cos^2 \theta_k \|\nabla f_k\|^2 < \infty.$$

This theorem guarantees global convergence.

1.4 Step Length Selection

We now consider techniques for finding a minimum of the one-dimensional function

$$\phi(\alpha) = f(x_k + \alpha p_k),$$

or for simply finding a step length α_k satisfying the condition described above.

We assume that p_k is a descent direction - that is, $\phi'(0) < 0$ - so that our search can be confined to positive values of α.

If f is a convex quadratic, $f(x) = \frac{1}{2}x^T Q x + b^T x + c$, its one-dimensional minimizer along the ray $x_k + \alpha p_k$ can be computed analytically and is given by

$$\alpha_k = -\frac{\nabla f_k^T p_k}{p_k^T Q p_k}.$$

If a general nonlinear function can be approximated as a quadratic model function, the model function can be optimized this way.

There are more advanced ways to select the step length α_k and step direction p_k.

For more details, please refer to ‘Numerical Optimization’ books.

2. Trust-Region Methods

A function f can be approximated as a quadratic model function m_k. We assume the quadratic model m_k at each iterate x_k matches the first- and second-order terms of the Taylor series of f around x_k.

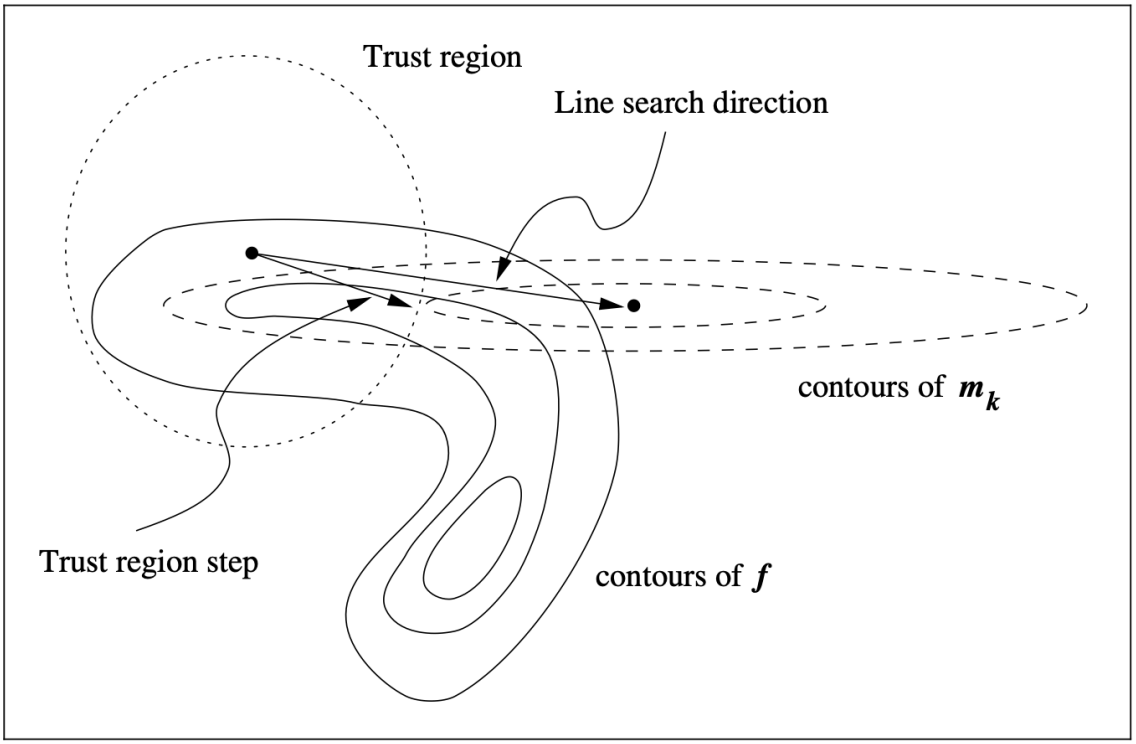
Specifically, we have

$$m_k(p) = f_k + \nabla f_k^T p + \frac{1}{2} p^T B_k p,$$

where $f_k = f(x_k)$, $\nabla f_k = \nabla f(x_k)$, and B_k is some symmetric matrix. The approximation error is $O(\|p\|^3)$ when B_k is equal to the true Hessian $\nabla^2 f(x_k)$, so this model is especially accurate when $\|p\|$ is small.

To obtain each step, we seek a solution to the subproblem

$$\min_{p \in R^n} m_k(p) = f_k + \nabla f_k^T p + \frac{1}{2} p^T B_k p \quad \text{s.t.} \|p\| \leq \Delta_k.$$



Trust-region and line search steps.

2.1 Outline of the Algorithm

Let's define the ratio given a step p_k,

$$\rho_k = \frac{f(x_k) - f(x_k + p_k)}{m_k(0) - m_k(p_k)};$$

the numerator is called the *actual reduction*, and the denominator is the *predicted reduction*.

Note that since the step p_k is obtained by minimizing the model m_k over a region that includes the step $p = 0$, the predicted reduction will always be non-negative. Thus if ρ_k is negative, the new objective value $f(x_k + p_k)$ is greater than the current value $f(x_k)$, so the step must be rejected.

On the other hand, if ρ_k is close to 1, there is good agreement between the model m_k and the function f over this step, so it is safe to expand the trust region for the next iteration. If ρ_k is positive but not close to 1, we do not alter the trust region, but if it is close to zero or negative, we shrink the trust region.

2.2 The Cauchy Point

Although in principle we are seeking the optimal solution to the subproblem above,
it is enough for global convergence purposes to find an approximate solution p_k that lies
within the trust region and gives a
sufficient reduction in the model. The sufficient reduction
can be quantified in terms of the Cauchy point, which we denote by p_k^{C} and define in terms
of the following simple procedure:

Cauchy Point Calculation

Find the vector p_k^{S} that solves a linear version of the subproblem:

$$p_k^{\mathrm{S}} = \underset{p \in R^n}{\arg\min}\, f_k + \nabla f_k^T p \qquad \mathrm{s.t.} \|p\| \leq \Delta_k;$$

That is simply

$$p_k^{\mathrm{S}} = -\frac{\Delta_k}{\|\nabla f_k\|}\nabla f_k.$$

Then, calculate the scalar $\tau_k > 0$ that minimizes $m_k(\tau p_k^{\mathrm{S}})$ subject to satisfying the trust-region bound, that is,

$$\tau_k = \arg\min_{\tau>0} m_k(\tau p_k^{\mathrm{S}}) \qquad \text{s.t.} \|\tau p_k^{\mathrm{S}}\| \le \Delta_k;$$

Set $p_k^{\mathrm{C}} = \tau_k p_k^{\mathrm{S}}$.

To obtain τ_k explicitly, we consider the cases of $\nabla f_k^T B_k \nabla f_k \le 0$ and $\nabla f_k^T B_k \nabla f_k > 0$ separately. For the former case, the function $m_k(\tau p_k^{\mathrm{S}})$ decreases monotonically with τ whenever $\nabla f_k \neq 0$, so τ_k is simply the largest value that satisfies the trust-region bound, namely, $\tau_k = 1$. For the latter case, $m_k(\tau p_k^{\mathrm{S}})$ is a convex quadratic in τ, so τ_k is either the unconstrained minimizer of this quadratic, $\|\nabla f_k\|^3/(\Delta_k \nabla f_k^T B_k \nabla f_k)$, or the boundary value 1, whichever comes first. In summary, we have

$$p_k^{\mathrm{C}} = -\tau_k \frac{\Delta_k}{\|\nabla f_k\|}\nabla f_k,$$

where

$$\tau_k = \begin{cases} 1 & \text{if } \nabla f_k^T B_k \nabla f_k \le 0; \\ \min(\|\nabla f_k\|^3/(\Delta_k \nabla f_k^T B_k \nabla f_k), 1) & \text{otherwise.} \end{cases}$$

There are more advanced ways to find the step p_k. For greater details, please refer to ‘Numerical Optimization’ books.

---

3. Conjugate Gradient

Suppose we want to solve the linear equation $Ax = b$ where A is an $n \times n$ matrix that is symmetric and positive definite. The solution is simple, $x = A^{-1}b$. However, calculating the inverse matrix A^{-1} is computationally heavy. The above linear system can equivalently be formulated as the minimization problem:

$$f(x) = \frac{1}{2}x^T Ax - b^T x$$

The derivative of the function f is:

$$f'(x) = \frac{1}{2}A^T x + \frac{1}{2}Ax - b = Ax - b,$$

which is because A is a symmetric matrix. We will find the solution x efficiently by applying the optimization methods to the function f instead of calculating A^{-1} directly.

3.1 Line Search (Steepest Descent) Method

Let's define some terms: The *error* $e_i = x_i - x$ is a vector that indicates how far we are from the solution. The *residual* $r_i = b - Ax_i$ indicates how far we are from the correct value of b. Then, $r_i = -Ae_i$ and $r_i = -f'(x_i)$. By line search method, the step length α_i can be chosen as:

$$\alpha = \frac{r_i^T r_i}{r_i^T A r_i}$$

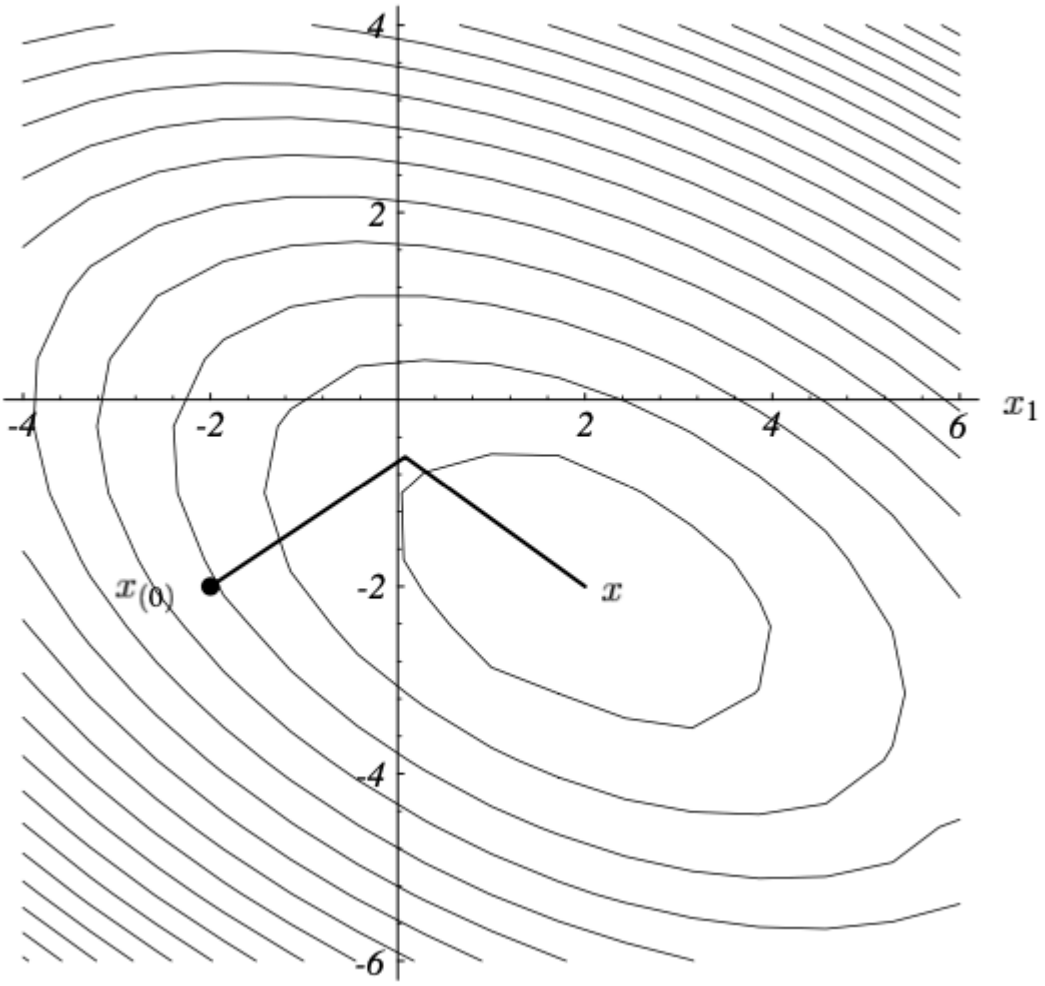
The step direction is chosen as the steepest descent:

$$p_i = -\nabla f_i = r_i$$

Therefore, the next step is taken as:

$$x_{i+1} = x_i + \alpha_i r_i.$$

Steepest Descent often finds itself taking steps in the same direction as earlier steps, which is inefficient. It would be better if we took a step and never step in that direction again.



The method of Conjugate Gradients

3.2 A-orthogonal and Conjugate Directions

Let's say two vectors d_i and d_j are *A-orthogonal*, or *conjugate* if

$$d_i^T A d_j = 0.$$

Instead of setting the search direction as gradient (line search), we pick a set of A-orthogonal search directions $d_0, d_1, ..., d_{n-1}$ and for each step we choose a point

$$x_{i+1} = x_i + \alpha_i d_i.$$

Step Lengths

To find the value of α_i, use the fact that e_{i+1} should be A-orthogonal to d_i, so that we need never step in the direction of d_i again. Actually, this orthogonality

condition is equivalent to finding the minimum point along the search direction d_i as in Steepest Descent. To see this, set the directional derivative to zero:

$$\frac{d}{d\alpha} f(x_{i+1}) = 0.$$

After some calculation, it can be easily shown that d_i and e_{i+1} are A-orthogonal, that is:

$$d_i^T A e_{i+1} = 0.$$

Since $e_{i+1} = e_i + \alpha_i d_i$ and $r_i = -Ae_i$,

$$d_i^T A(e_i + \alpha_i d_i) = 0$$

$$\begin{aligned} \alpha_i &= -\frac{d_i^T A e_i}{d_i^T A d_i} \\ &= \frac{d_i^T r_i}{d_i^T A d_i}. \end{aligned}$$

Then, we expect to find the exact solution within n steps.

Gram-Schmidt Conjugation

All that is needed is a set of A-orthogonal search directions $\{d_i\}$. Set $d_0 = u_0$ and for $i > 0$, set

$$d_i = u_i + \sum_{k=0}^{i-1} \beta_{ik} d_k,$$

where the β_{ik} are defined for $i > k$ and n linearly independent vectors $\{u_i\}$. To find their values, we take the inner product of two vectors d_i and d_j $(i > j)$ with the kernel (or metric) A, that is:

$$d_i^T A d_j = u_i^T A d_j + \sum_{k=0}^{i-1} \beta_{ij} d_k^T A d_j$$

By the A-orthogonality:

$$0 = u^T A d_j + \beta_{ij} d_j^T A d_j$$

$$\beta_{ij} = -\frac{u_i^T A d_j}{d_j^T A d_j}$$

This process is called the *conjugate Gram-Schmidt process*.

If we construct d_i in this way, the d_i *component* of the error term e_j ($d_i^T e^j$) is zero when $i < j$.

By taking the inner product of d_i and r_j, it can be proved that r_j is orthogonal to $u_0, ..., u_{j-1}$:

This means that two vectors r_i and r_j are orthogonal if $i \neq j$.

3.3 The Method of Conjugate Gradients

With the fact that $\{r_i\}$ is a set of orthogonal vectors, we can construct d_i from r_i - that is, we can set $u_i = r_i$.

Given $r_{j+1} = -Ae_{j+1} = -A(e_j + \alpha_j d_j) = r_j - \alpha_j A d_j$, take the inner product of r_i and r_{j+1},

$$r_i^T r_{j+1} = r^T r_j - \alpha_j r_i^T A d_j$$

$$r_i^T A d_j = \begin{cases} \frac{1}{\alpha_i} r_i^T r_i & i = j \\ -\frac{1}{\alpha_{i-1}} r_i^T r_i & i = j+1 \\ 0 & \text{otherwise} \end{cases}$$

$$\therefore \beta_{ij} = \begin{cases} \frac{1}{\alpha_{i-1}} \frac{r_i^T r_i}{d_{i-1}^T A d_{i-1}} & i = j+1 \\ 0 & i > j+1 \end{cases}$$

To simplify the notation, let's define $\beta_i = \beta_{i,i-1}$. Then,

$$\beta_i = \frac{1}{\alpha_{i-1}} \frac{r_i^T r_i}{d_{i-1}^T A d_{i-1}} = \frac{r_i^T r_i}{d_{i-1}^T r_{i-1}} = \frac{r^T r_i}{r_{i-1}^T r_{i-1}}.$$

For the last equation, the orthogonality of the r vectors is used.

Let's put it all together. The method of Conjugate Gradients is:

$$d_0 = r_0 = b - Ax_0$$

$$\alpha_i = \frac{r_i^T r_i}{d_i^T A d_i}$$

$$x_{i+1} = x_i + \alpha_i d_i$$

$$r_{i+1} = r_i - \alpha_i A d_i$$

$$\beta_{i+1} = \frac{r_{i+1}^T r_{i+1}}{r_i^T r_i}$$

$$d_{i+1} = r_{i+1} + \beta_{i+1} d_i$$

3.4 Nonlinear Conjugate Gradient

In nonlinear CG, there are several ways to obtain the value of β. Two choices are the Fletcher-Reeves formula and the Polak-Ribiere formula:

$$\beta_{i+1}^{FR} = \frac{r_{i+1}^T r_{i+1}}{r_i^T r_i}, \qquad \beta_{i+1}^{PR} = \max\{\frac{r_{i+1}^T (r_{i+1} - r_i)}{r_i^T r_i}, 0\}.$$

The outline of the nonlinear CG method is:

$$d_0 = r_0 = -f'(x_0)$$

$$\text{Find } \alpha_i \text{ that minimizes } f(x_i + \alpha_i d_i)$$

$$x_{i+1} = x_i + \alpha_i d_i$$

$$r_{i+1} = -f'(x_{i+1})$$

$$\text{Find } \beta_{i+1} = \beta_{i+1}^{FR} \text{ or } \beta_{i+1} = \beta_{i+1}^{PR}$$

$$d_{i+1} = r_{i+1} + \beta_{i+1} d_i$$

α_i can be found by Newton's Method or Quasi-Newton Method and so on.