Possible Future Research

Trever Hallock

March 16, 2017

we can cite [1] [2] [3] [4] [5] [6] [7] [8] [9] [10] [11]

1 Review of Model Based DFO methods

2 Optimization with no derivatives

- Evaluating $f(x)$ may involve running a simulation
- The runtime of $f(x)$ may mean that typical finite difference methods are too expensive
- The derivative free philosophy is to avoid function evaluations

3 Derivatives are convenient

- Quadratic convergence in line search methods require second order derivatives
- They appear in conditions for convergence results (Wolf, Armijo, or Goldstein for line search)
- Stopping criteria are based on optimality conditions, which frequently involve derivatives

3.1

Consider the following optimization problem

$$\begin{aligned} \underset{x}{\text{minimize}} \quad & f(x, S(x)) \\ \text{subject to} \quad & g_i(x, S(x)) \leq 0,\ i = 1, \dots, m \\ & h_i(x, S(x)) = 0,\ i = 1, \dots, n \end{aligned}$$

This gives rise to different classes of derivative free optimization based on f, g_i, and h_i:

- They may be noisy
- Derivatives of g_i and h_i may be known
- g_i and h_i can sometimes be "Hidden" constraints
- Sometimes the functions g_i and h_i are only found by evaluating $S(x)$
- Many DFO methods simply let $f(x, S(x)) = S(x)$

A slightly more general form is:

$$\begin{aligned} \underset{x}{\text{minimize}} \quad & f(x, y, z) \\ \text{subject to} \quad & g_i(x, y, z) \leq 0 \quad \forall i = 1, \dots, m_1 \\ & h_i(x, y, z) = 0, \quad \forall i = 1 \dots, m_2 \\ & y = d(z) \end{aligned}$$

4 Approaches

- Finite difference methods
  - $\nabla f(x) \approx (\frac{f(x+he_i)-f(x-he_i)}{2h})_i$ for some h
  - The number of function evaluations can grow large quickly
- Direct search methods
  - Coordinate descent and Pattern based
  - Nelder Mead
  - Do not use derivatives: robust but ignore useful information
- Model-based methods
  - This is what we will discuss

5 The Derivative Trust Region Method

- Interpolate or regress model functions onto a sample
- Minimize the model function over a trust region
- Adjust trust region, possibly by testing the model's accuracy

6 More details

1. Define $m_k(x) = f(x^{(k)}) + \nabla f(x^{(k)})^T(x - x^{(k)}) + \frac{1}{2}(x - x^{(k)})^T\nabla^2 f(x^{(k)})(x - x^{(k)})$
   - $\nabla f(x^{(k)})$ and $\nabla^2 f(x^{(k)})$ must be approximated
   - There are geometric properties of the sample set that must be satisfied
2. if $\nabla m_k(x) < t$ stop
3. Solve the Trust region subproblem: $s^{(k)} = \arg\min_{s \in B_(x^{(k)}, \Delta_k)} m_k(x^{(k)} + s)$
4. Test for improvement
   - $\rho_k = \frac{f(x^{(k)})-f(x^{(k)}+s^{(k)})}{m_k(x^{(k)})-m_k(x^{(k)}+s^{(k)})}$
   - If ρ is small, $x^{(k+1)} = x^{(k)}$ (reject) and decrease radius
   - If ρ is intermediate, $x^{(k+1)} = x^{(k)} + s^{(k)}$ (accept) and decrease radius
   - If ρ is large, $x^{(k+1)} = x^{(k)} + s^{(k)}$ (accept) and increase radius

7 Possible future work

8 Future Directions

- Convert classical NLP Algorithms to DFO
- Different Goals
- Parallelization
- Explore different model functions
- Providing structure to the optimization program

9 NLP Methods

We can convert classical algorithms to DFO

- Take an existing constrained Nonlinear programming method
  - Filter method
  - Active Set
  - Augmented Lagrangian
  - Interior Point
  - Line search
- Replace derivatives of f, g_i, h_i with derivatives of a model function

10 Goals

We could consider an objective function with a runtime that varies based on the position.

11 Parallelization

1. parallelization of the function and/or the derivative evaluations in the algorithm
2. parallelization of linear algebra kernels
3. modifications of the basic algorithms which increase the degree of intrinsic parallelism, for instance, by performing multiple function and/or derivative evaluations

The third item is particularily interesting.

12 Miscellaneous

- Model Functions Radial basis functions
- Problem Structure Specify the structure of f, g_i, or h_i
- Machine Learning Applying DFO techniques to algorthms that currently use random sampling to tune parameters

13 My Algorithm

14 Problem

We began with a line search filter method, however we found that this had several drawbacks:

- Trust regions arise naturally within derivative free algorithms
- Line search algorithms exploit how much easier finding a descent direction is than solving the trust region subproblem, but this saved computation is not as useful in DFO
- As the algorithm backtracks on the step length, the trust region must be reduced, as the models are accurate over a region rather than at a single point

15 Algorithm in other paper

16 problem statement

We consider problems of the form

$$\begin{aligned} \min_x & f(x) \\ & g(x) \leq 0 \\ & h(x) = 0 \end{aligned}$$

where $f : R^n \to R$, $g : R^n \to R^{m_1}$ and $h : R^n \to R^{m_2}$.

We assume that all functions are derivative free: all functions f, g, h are evaluated by a single call to a black box function $d(x) = (f(x), g(x)^T, h(x)^T)^T$.

We work within a trust region, sequential quadratic programming framework that uses a filter method introduced by Fletcher.

We first compute an interpolation set poised for regressing a set of model functions, which we choose to be quadratic functions. Although we have function values for enough points to construct model functions of the same order as used for the objective, we model the feasible with only the linear constraints.

The core of the algorithm revolves around

17 Step decomposition

18 Compatibility

19 The filter

20 f-type versus θ-type

21 restoration step

22

23 What I think the algorithm should be

24 Pictures of convergence...

Algorithm 1 Filter Trust Region Search

procedure TRUST REGION FILTER
  initialize
  $k = 0$
  choose an x_0
  while $k < maxit$ **do**
`main loop:`
    ensure poisedness, possibly adding points to the model
    Compute $m_k, g_k = \nabla m_k(x_k), c_k, A_k, f_k = f(x_k), \mathcal{A}, \theta_k$
    Solve:

$$\begin{aligned} \nabla^2 m_k(x_k)d + A_k^T\lambda &= g_k \\ A_k d \qquad\quad &= c_k \end{aligned}$$

    $H_k \leftarrow \nabla^2 m_k(x_k) + \sum_i \lambda_i \nabla^2 c_{ik}$
    $\chi_k \leftarrow |\min_t\{\langle g_k + H_k n_k, t\rangle | A_{eq}t = 0 \wedge c_{ineq} + A_{ineq}t \leq 0 \wedge \|t\| \leq 1\}|$
    if constraint violation $= 0 \wedge \chi = 0$ **then**
      if $tol < \Delta_k$ **then**
        reduce Δ: $\Delta_{k+1} \leftarrow$ some $\in [\gamma_0\Delta_k, \gamma_1\Delta_k]$
        $k \leftarrow k + 1$
        go to `main loop`
      success
    $n_k \leftarrow \arg\min_n\{\|n\|^2 | c_{eq} + A_{eq}n = 0 \wedge c_{ineq} + A_{ineq}n \leq 0 \wedge \|n\|^2 \leq \Delta_k\}^2$
    if Feasible region $\neq \emptyset \wedge \|n\| \leq \kappa_\Delta \Delta_k \min\{1, \kappa_\mu \Delta_k^\mu\}$ **then**
      $t_k \leftarrow \arg\min_t\{(g_n + H_k n_k)^T t + \frac{1}{2}t^T H_k t | c_{eq} + A_{eq}t = 0 \wedge c_{ineq} + A_{ineq}t \leq 0 \wedge \|s\| \leq \Delta_k\}$
      $s_k \leftarrow t_k + n_k$
      if $m_k(x_k) - m_k(x_k + s_k) \geq \kappa_\theta \theta_k^\psi$ **then**
        add x_k to filter
        reduce Δ: $\Delta_{k+1} \leftarrow$ some $\in [\gamma_0\Delta_k, \gamma_1\Delta_k]$
        go to `main loop`
      // Here we evaluate new c and f at $x_k + s_k$
      if $x_k + s_k$ is acceptable: $\theta(x_k + s_k) \leq (1 - \gamma_\theta)\theta' \vee f(x_k + s_k) \leq f' - \gamma_\theta\theta' \forall (f', \theta') \in$ Filter **then**
        $\rho = \frac{f(x_k) - f(x_k + s_k)}{m_k(x_k) - m_k(x_k + s_k)}$
        if $\rho < \eta_1$ **then**
          reduce Δ: $\Delta_{k+1} \leftarrow$ some $\in [\gamma_0\Delta_k, \gamma_1\Delta_k]$
          $k \leftarrow k + 1$
          go to `main loop`
        else if $\rho > \eta_2$ **then**
          if $\|s\| < \frac{\Delta_k}{2}$ **then**
            reduce Δ: $\Delta_{k+1} \leftarrow$ some $\in [\gamma_0\Delta_k, \gamma_1\Delta_k]$
          else
            increase Δ: $\Delta_{k+1} \leftarrow$ some $\in [\Delta_k, \gamma_2\Delta_k]$
        $x_{k+1} \leftarrow x_k + s_k$
      else
        reduce Δ: $\Delta_{k+1} \leftarrow$ some $\in [\gamma_0\Delta_k, \gamma_1\Delta_k]$
        $k \leftarrow k + 1$
        go to `main loop`
    else
      add x_k to filter
      compute new r (restoration step) and Δ
      if impossible to restore **then fail**
      $x_{k+1} \leftarrow x_k + r$
    $k \leftarrow k + 1$

References

[1] J. Doe, *The Book without Title*. Dummy Publisher, 2100.

[2] J. P. Eason and L. T. Biegler, "A trust region filter method for glass box/black box optimization," *AIChE Journal*, vol. 62, no. 9, pp. 3124–3136, 2016.

[3] R. Fletcher, S. Leyffer, R. Fletcher, and S. Leyffer, "A brief history of filter methods," tech. rep., 2006.

[4] R. Brekelmans, L. Driessen, H. Hamers, and D. den Hertog, "Constrained optimization involving expensive function evaluations: A sequential approach," *European Journal of Operational Research*, vol. 160, no. 1, pp. 121 – 138, 2005. Applications of Mathematical Programming Models.

[5] J. Nocedal and Y. Yuan, "Combining trust region and line search techniques,"

[6] A. Wächter and L. T. Biegler, "Line search filter methods for nonlinear programming: Motivation and global convergence," *SIAM Journal on Optimization*, vol. 16, no. 1, pp. 1–31, 2005.

[7] A. Wächter and L. T. Biegler, "Line search filter methods for nonlinear programming: Local convergence," *SIAM Journal on Optimization*, vol. 16, no. 1, pp. 32–48, 2005.

[8] A. R. Conn, K. Scheinberg, and L. N. Vicente, *Introduction to Derivative-Free Optimization*. Philadelphia, PA, USA: Society for Industrial and Applied Mathematics, 2009.

[9] R. Sampaio and L. Toint, "A trust-funnel method for nonlinear optimization problems with general nonlinear constraints and its application to derivative-free optimization, most information in this cite is wrong," *NAXYS Namur Center for Complex Systems*, vol. 61, no. 5000, pp. 1–31, 2015.

[10] N. I. M. Gould and P. L. Toint, "Nonlinear programming without a penalty function or a filter," *Mathematical Programming*, vol. 122, no. 1, pp. 155–196, 2010.

[11] P. T. Boggs and J. W. Tolle, "Sequential quadratic programming for large-scale nonlinear optimization," *Journal of Computational and Applied Mathematics*, vol. 124, no. 12, pp. 123 – 137, 2000. Numerical Analysis 2000. Vol. IV: Optimization and Nonlinear Equations.