Scalable Derivative-Free Optimization for Nonlinear Least-Squares Problems

With Coralia Cartis & Tyler Ferguson (Oxford)

Lindon Roberts, Mathematical Sciences Institute, ANU (`lindon.roberts@anu.edu.au`)

5th Workshop on Optimisation, Metric Bounds, Approximation and Transversality
1 December 2020

Outline

1. Derivative-free optimization for least-squares problems
2. Scalability bottleneck
3. Sketching techniques
4. Numerical results

Derivative-Free Optimization

$$\min_{x \in \mathbb{R}^d} f(x)$$

- Objective f nonlinear, nonconvex, structure unknown

Derivative-Free Optimization

$$\min_{x \in \mathbb{R}^d} f(x)$$

- Objective f nonlinear, nonconvex, structure unknown
- Standard methods locally approximate f by quadratic models (e.g. Taylor series)
- How to calculate derivatives of f to build model?
  - Write code by hand
  - Finite differences
  - Algorithmic differentiation [aka backpropagation]

Derivative-Free Optimization

$$\min_{x \in \mathbb{R}^d} f(x)$$

- Objective f nonlinear, nonconvex, structure unknown
- Standard methods locally approximate f by quadratic models (e.g. Taylor series)
- How to calculate derivatives of f to build model?
  - Write code by hand
  - Finite differences
  - Algorithmic differentiation [aka backpropagation]
- Difficulties when function evaluation is
  - 'Black-box'
  - Noisy
  - Computationally expensive

Derivative-Free Optimization

$$\min_{x\in\mathbb{R}^d} f(x)$$

- Objective f nonlinear, nonconvex, structure unknown
- Standard methods locally approximate f by quadratic models (e.g. Taylor series)
- How to calculate derivatives of f to build model?
  - Write code by hand
  - Finite differences
  - Algorithmic differentiation [aka backpropagation]
- Difficulties when function evaluation is
  - 'Black-box'
  - Noisy
  - Computationally expensive
- Alternative — derivative-free optimization (DFO) [aka "zero-order methods"]
  - Applications in finance, climate, image analysis, data science, engineering, ...

Model-Based DFO — Basic Ideas

Many approaches: model-based, direct search, Nelder-Mead, ...

- Classically (e.g. Newton's method),

$$f(x_k + s) \approx m_k(s) = f(x_k) + \nabla f(x_k)^T s + \frac{1}{2} s^T \nabla^2 f(x_k) s$$

Model-Based DFO — Basic Ideas

Many approaches: model-based, direct search, Nelder-Mead, ...

- Classically (e.g. Newton's method),

$$f(x_k + s) \approx m_k(s) = f(x_k) + \nabla f(x_k)^T s + \frac{1}{2} s^T \nabla^2 f(x_k) s$$

- Instead, approximate

$$f(x_k + s) \approx m_k(s) = f(x_k) + g_k{}^T s + \frac{1}{2} s^T H_k s$$

Model-Based DFO — Basic Ideas

Many approaches: model-based, direct search, Nelder-Mead, ...

- Classically (e.g. Newton's method),

$$f(x_k + s) \approx m_k(s) = f(x_k) + \nabla f(x_k)^T s + \frac{1}{2} s^T \nabla^2 f(x_k) s$$

- Instead, approximate

$$f(x_k + s) \approx m_k(s) = f(x_k) + g_k^T s + \frac{1}{2} s^T H_k s$$

- Find g_k and H_k without using derivatives: interpolate f over a set of points
- Geometry of points good $\implies$ interpolation model accurate $\implies$ convergence

[Conn, Powell, Scheinberg, Vicente, ...]

DFO for Least-Squares — Basic Framework

$$\min_{x\in\mathbb{R}^d} f(x) = \frac{1}{2}\|r(x)\|_2^2, \qquad r(x) \in \mathbb{R}^n$$

Classical Gauss-Newton **Derivative-Free Gauss-Newton**

DFO for Least-Squares — Basic Framework

$$\min_{x\in\mathbb{R}^d} f(x) = \frac{1}{2}\|r(x)\|_2^2, \qquad r(x) \in \mathbb{R}^n$$

Classical Gauss-Newton

- Linearize r at x_k using Jacobian

$$r(x_k + s) \approx M_k(s) = r(x_k) + J(x_k)s$$

Derivative-Free Gauss-Newton

DFO for Least-Squares — Basic Framework

$$\min_{x\in\mathbb{R}^d} f(x) = \frac{1}{2}\|r(x)\|_2^2, \qquad r(x) \in \mathbb{R}^n$$

Classical Gauss-Newton

- Linearize r at x_k using Jacobian

$$r(x_k + s) \approx M_k(s) = r(x_k) + J(x_k)s$$

Derivative-Free Gauss-Newton

- Jacobian not available: use

$$M_k(s) = r(x_k) + J_k s$$

- Find J_k by interpolation — maintain a cloud of points which moves towards solution (with good geometry)

DFO for Least-Squares — Basic Framework

$$\min_{x \in \mathbb{R}^d} f(x) = \frac{1}{2}\|r(x)\|_2^2, \qquad r(x) \in \mathbb{R}^n$$

Classical Gauss-Newton

- Linearize r at x_k using Jacobian

$$r(x_k + s) \approx M_k(s) = r(x_k) + J(x_k)s$$

Derivative-Free Gauss-Newton

- Jacobian not available: use

$$M_k(s) = r(x_k) + J_k s$$

- Find J_k by interpolation — maintain a cloud of points which moves towards solution (with good geometry)

In both cases, get a local quadratic model (with approximate Hessian)

$$f(x_k + s) \approx m_k(s) = \frac{1}{2}\|M_k(s)\|_2^2$$

DFO for Least-Squares — Algorithm

Implement in trust-region method:

1. Build interpolation model

$$f(x_k + s) \approx m_k(s) := \frac{1}{2}\|M_k(s)\|_2^2.$$

2. Minimize model inside trust region

$$s_k = \operatorname*{arg\,min}_{s \in \mathbb{R}^d} m_k(s) \quad \text{s.t.} \quad \|s\|_2 \leq \Delta_k.$$

3. Evaluate $f(x_k + s_k)$, check sufficient decrease, select x_{k+1} and Δ_{k+1}
4. Update interpolation set: add $x_k + s_k$ and move points to ensure good geometry (if needed)
   $\leftarrow$ requires calculation of Lagrange polynomials

Implemented in DFO-LS package (Github: `numericalalgorithmsgroup/dfols`*)*

(Also have software for general objectives using quadratic interpolation)

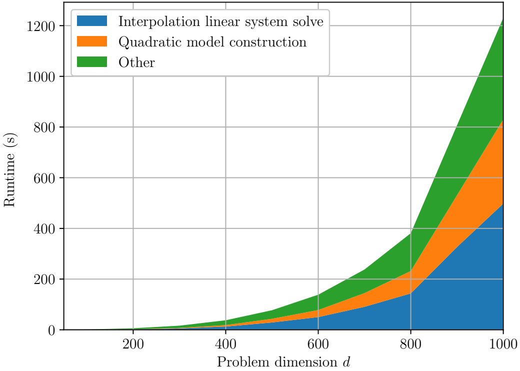
Scalability

- DFO methods are well-known not to scale well (i.e. n or d large)
  - e.g. data science, weather forecasting/data assimilation, ...

Where is the issue for model-based DFO?

Scalability

Runtime of DFO-LS on generalized Rosenbrock function ($n = 2d$):



Per-iteration linear algebra costs:

- Interpolation linear system $= \mathcal{O}(d^3 + nd^2)$
- Form quadratic model $= \mathcal{O}(nd^2)$
- Subproblem, geometry, ... $= \mathcal{O}(d^3)$

(d variables, n residuals)

Problem

Goal

Can we construct a method which is more efficient in terms of runtime?

Key idea: dimensionality reduction in n

- Here, try to improve performance in the case of 'big data' ($n \to \infty$)
- (also studying dimensionality reduction in d-dimensional variable space in other work)
- Use ideas from randomized numerical linear algebra (specifically sketching)

Sketching

Inspired by randomized methods for linear least-squares

$$\min_{x \in \mathbb{R}^d} \|Ax - b\|_2^2, \qquad A \in \mathbb{R}^{n \times d} \text{ full rank, } n \gg d.$$

Standard methods (e.g. QR factorization) have cost $\mathcal{O}(nd^2)$.

Instead, generate random matrix $S \in \mathbb{R}^{m \times n}$ ($m \ll n$) and solve the smaller $m \times d$ problem

$$\min_{x' \in \mathbb{R}^d} \|(SA)x' - (Sb)\|_2^2.$$

Good choices of S?

- Easy to construct SA and Sb (needs to be faster than $\mathcal{O}(nd^2)$!)
- Solution to sketched problem close to solution of original problem
  - S approximately preserves inner products in $\mathrm{col}(A) \oplus \mathrm{span}\{b\}$

Sketching

Common choices for S include:

- Gaussian matrix (each entry iid normal)
- Subsampling matrix (each row is random coordinate vector)
- Hashing matrix (each column has a small number of randomly-placed ± 1 entries)

The last two choices are sparse, so matrix multiplication is cheap. Typical results look like...

Theorem (Woodruff, 2014)

Suppose S is a hashing matrix with $m = \mathcal{O}(d^2/\epsilon^2 \cdot \mathrm{poly}(\log(d/\epsilon)))$, and x' is the minimizer of the sketched LLS problem. *[Note: m is independent of n]*

Then x' can be found in $\mathcal{O}(\mathrm{nnz}(A) + \mathrm{poly}(d/\epsilon))$ time and

$$\|Ax' - b\|_2 \leq (1+\epsilon) \min_{x \in \mathbb{R}^d} \|Ax - b\|_2.$$

with probability at least 0.99.

Use of Sketching

Sketching ideas are gaining lots of attention, particularly in algorithms for data science:

- Linear least-squares [Drineas et al (2006), Clarkson & Woodruff (2017)]
- Matrix factorizations (e.g. randomized SVD) [Mahoney (2011), Halko, Martinsson & Tropp (2011)]
- BFGS [Gower, Goldfarb, & Richtárik (2016)]
- Newton's Method [Roosta-Khorasani & Mahoney (2019), Berahas, Bollapragada & Nocedal (2020)]
- Nonlinear least-squares [Ergen, Candès & Pilanci (2019)]

and more...

Question

Does the success of sketching apply to DFO?

Sketching for DFO I

We know that DFO-LS is slow because of the cost of solving the interpolation system.

Question

Can sketching be used to reduce the linear algebra cost of DFO?

Interpolation system:

Given iterate x_k and points $y_1, \ldots, y_d$, build the linear model $M_k(s) = r(x_k) + J_k s$ by solving

$$\begin{bmatrix} (y_1 - x_k)^T \\ \vdots \\ (y_d - x_k)^T \end{bmatrix} J_k^T = \begin{bmatrix} (r(y_1) - r(x_k))^T \\ \vdots \\ (r(y_1) - r(x_k))^T \end{bmatrix},$$

with cost $\mathcal{O}(d^3 + nd^2)$ [factorization plus n back-substitutions].

Sketching for DFO II

Apply sketching to the interpolation system:

At each iteration, generate random S_k and solve the smaller system ($m \ll n$ RHS)

$$\begin{bmatrix} (y_1 - x_k)^T \\ \vdots \\ (y_d - x_k)^T \end{bmatrix} (S_k J_k)^T = \begin{bmatrix} (r(y_1) - r(x_k))^T \\ \vdots \\ (r(y_1) - r(x_k))^T \end{bmatrix} S_k^T.$$

Then, we get a linear model for $S_k r(\cdot) : \mathbb{R}^d \to \mathbb{R}^m$:

$$S_k r(x_k + s) \approx \widetilde{M}_k(s) = S_k r(x_k) + (S_k J_k)s,$$

giving the local quadratic model

$$f(x_k + s) \approx \widetilde{m}_k(s) = \frac{1}{2}\|\widetilde{M}_k(s)\|^2.$$

Sketching for DFO III

Question

Does this reduce the linear algebra cost of DFO?

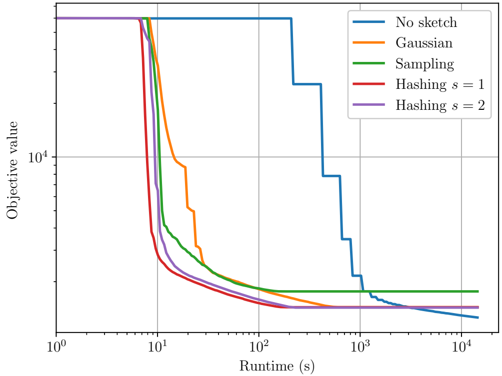
Exactly the same DFO algorithm, but using sketched interpolation system:

	Original	Gaussian	Sampling	Hashing
Form S_k	—	$\mathcal{O}(mn)$	$\mathcal{O}(m)$	$\mathcal{O}(nd)$
Form sketched system	—	$\mathcal{O}(mnd)$	$\mathcal{O}(nd)$	$\mathcal{O}(nd)$
Solve interpolation system	$\mathcal{O}(d^3 + nd^2)$	$\mathcal{O}(d^3 + md^2)$	$\mathcal{O}(d^3 + md^2)$	
Form quadratic model	$\mathcal{O}(nd^2)$	$\mathcal{O}(md^2)$	$\mathcal{O}(md^2)$	
Subproblem, geometry, ...	$\mathcal{O}(d^3)$	$\mathcal{O}(d^3)$	$\mathcal{O}(d^3)$	
Total (if $n \gg d$)	$\boldsymbol{\mathcal{O}(nd^2)}$	$\boldsymbol{\mathcal{O}(mnd)}$	$\boldsymbol{\mathcal{O}(nd + d^3 + md^2)}$	

In the 'big data' regime ($n \to \infty$), sampling/hashing reduces cost from $\mathcal{O}(nd^2)$ to $\mathcal{O}(nd)$.

Numerical Results

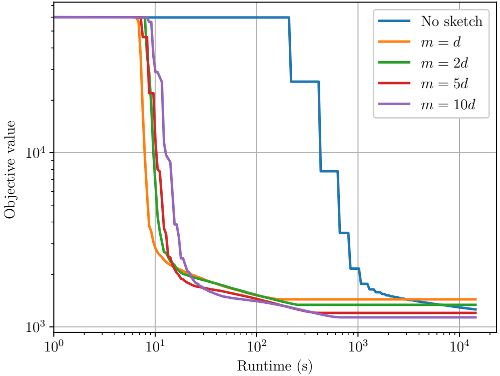
Compare on a large problem from the CUTEst collection (MNISTS0 with $d = 494$, $n = 60\,000$)



Objective value vs. runtime ($m = d$ for all sketches, 4hr timeout)

Numerical Results

What difference does m make? (MNISTS0 with $d = 494$, $n = 60\,000$)



Objective value vs. runtime (hashing $s = 1$ for all, 4hr timeout)

Conclusion & Future Work

Conclusions

- Model construction cost a key barrier to scalability of model-based DFO
- Modify model construction by using sketched interpolation system
- In big data regime, linear algebra cost reduces from $\mathcal{O}(nd^2)$ to $\mathcal{O}(nd)$
- Gives improved runtime performance in practice

ICML workshop paper available at arXiv:2007.13243

Future Work

- Convergence guarantees
- Adaptive sketching (e.g. vary m as algorithm progresses)
- Dimensionality reduction in variable space (coming soon, see AustMS talk...)

References

Coralia Cartis, Tyler Ferguson, and Lindon Roberts.
Scalable derivative-free optimization for nonlinear least-squares problems.
In *Workshop on "Beyond first-order methods in ML systems" at the 37th International Conference on Machine Learning*, 2020.

Coralia Cartis, Jan Fiala, Benjamin Marteau, and Lindon Roberts.
Improving the flexibility and robustness of model-based derivative-free optimization solvers.
ACM Transactions on Mathematical Software, 45(3):32:1–32:41, 2019.

Coralia Cartis and Lindon Roberts.
A derivative-free Gauss-Newton method.
Mathematical Programming Computation, 11(4):631–674, 2019.