Randomised Subspace Methods for Scalable Derivative-Free Optimisation

Joint work with Coralia Cartis (Oxford), Clément Royer (Paris-Dauphine PSL), Warren Hare (UBC)

Lindon Roberts, University of Sydney (`lindon.roberts@sydney.edu.au`)

Applied Mathematics Seminar, UNSW Sydney
19 September 2024

Further Reading

This talk is based on:

- C. Cartis & LR, Scalable subspace methods for derivative-free nonlinear least-squares optimization, *Mathematical Programming* 199:1–2 (2023), pp. 461–524.
- LR & C. W. Royer, Direct search based on probabilistic descent in reduced spaces, *SIAM Journal on Optimimization* 33:4 (2023), pp. 3057–3082.
- W. Hare, LR & C. W. Royer, Expected decrease for derivative-free algorithms using random subspaces, *Mathematics of Computation*, accepted, 2024.

Software packages are available on Github.

Outline

1. **Introduction to derivative-free optimisation (DFO)**
2. Subspace DFO methods
3. Average-case analysis

Optimisation in Data Science

Optimisation is fundamental to data science. For example, to fit a predictive model

$$\boldsymbol{v} \approx m(\boldsymbol{u}, \boldsymbol{x})$$

(e.g. linear/nonlinear regression, neural networks) we usually have training data $(\boldsymbol{u}_i, \boldsymbol{v}_i)$ and solve the empirical risk minimisation problem

$$\min_{\boldsymbol{x}} f(\boldsymbol{x}) = \frac{1}{N}\sum_{i=1}^{N} \ell(\boldsymbol{v}_i, m(\boldsymbol{u}_i, \boldsymbol{x})),$$

for some loss function ℓ, for example $\ell(\boldsymbol{v}_1, \boldsymbol{v}_2) = \|\boldsymbol{v}_1 - \boldsymbol{v}_2\|^2$.

This is a well-studied mathematical problem (and relevant to many other disciplines too).

Gradient Descent

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

For any $\boldsymbol{x}$, the vector $\nabla f(\boldsymbol{x})$ points in the direction of fastest ascent (locally).

Gradient Descent

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

For any $\boldsymbol{x}$, the vector $\nabla f(\boldsymbol{x})$ points in the direction of fastest ascent (locally).

Gradient descent iterates to a solution by stepping in the $-\nabla f$ direction

$$\boldsymbol{x}_{k+1} = \boldsymbol{x}_k - \alpha_k \nabla f(\boldsymbol{x}_k)$$

Gradient Descent

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

For any $\boldsymbol{x}$, the vector $\nabla f(\boldsymbol{x})$ points in the direction of fastest ascent (locally).

Gradient descent iterates to a solution by stepping in the $-\nabla f$ direction

$$\boldsymbol{x}_{k+1} = \boldsymbol{x}_k - \alpha_k \nabla f(\boldsymbol{x}_k)$$

Theorem

Suppose $f \in C^2(\mathbb{R}^n)$ bounded below, with $\|\nabla^2 f(\boldsymbol{x})\|_2 \leq H_{\max}$ everywhere.

If $\alpha_k = 1/H_{\max}$ for all k, then $\lim_{k\to\infty} \nabla f(\boldsymbol{x}_k) = \boldsymbol{0}$.

Gradient Descent

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

For any $\boldsymbol{x}$, the vector $\nabla f(\boldsymbol{x})$ points in the direction of fastest ascent (locally).

Gradient descent iterates to a solution by stepping in the $-\nabla f$ direction

$$\boldsymbol{x}_{k+1} = \boldsymbol{x}_k - \alpha_k \nabla f(\boldsymbol{x}_k)$$

Theorem

Suppose $f \in C^2(\mathbb{R}^n)$ bounded below, with $\|\nabla^2 f(\boldsymbol{x})\|_2 \leq H_{\max}$ everywhere.

If $\alpha_k = 1/H_{\max}$ for all k, then $\lim_{k\to\infty} \nabla f(\boldsymbol{x}_k) = \boldsymbol{0}$.

If N large, often average over random subsets to get random approximations $\boldsymbol{g}_k \approx \nabla f(\boldsymbol{x}_k) \quad \to$ *stochastic gradient descent.*

Gradient Descent: Practicalities

$$\boldsymbol{x}_{k+1} = \boldsymbol{x}_k - \alpha_k \nabla f(\boldsymbol{x}_k)$$

- How to calculate derivatives of f?
  - Write code by hand
  - Finite differences, $f'(x) \approx \frac{f(x+h)-f(x)}{h}$
  - Algorithmic differentiation/backpropagation

Gradient Descent: Practicalities

$$\boldsymbol{x}_{k+1} = \boldsymbol{x}_k - \alpha_k \nabla f(\boldsymbol{x}_k)$$

- How to calculate derivatives of f?
  - Write code by hand
  - Finite differences, $f'(x) \approx \frac{f(x+h)-f(x)}{h}$
  - Algorithmic differentiation/backpropagation
- Impractical if function evaluation is black-box, computationally expensive or noisy

Gradient Descent: Practicalities

$$\boldsymbol{x}_{k+1} = \boldsymbol{x}_k - \alpha_k \nabla f(\boldsymbol{x}_k)$$

- How to calculate derivatives of f?
  - Write code by hand
  - Finite differences, $f'(x) \approx \frac{f(x+h)-f(x)}{h}$
  - Algorithmic differentiation/backpropagation
- Impractical if function evaluation is black-box, computationally expensive or noisy
- How to pick stepsize/learning rate?
  - Calculate $H_{\max}$
  - Hyperparameter tuning
  - Adaptive procedures (e.g. linesearch)

Gradient Descent: Practicalities

$$\boldsymbol{x}_{k+1} = \boldsymbol{x}_k - \alpha_k \nabla f(\boldsymbol{x}_k)$$

- How to calculate derivatives of f?
  - Write code by hand
  - Finite differences, $f'(x) \approx \frac{f(x+h)-f(x)}{h}$
  - Algorithmic differentiation/backpropagation
- Impractical if function evaluation is black-box, computationally expensive or noisy
- How to pick stepsize/learning rate?
  - Calculate $H_{\max}$
  - Hyperparameter tuning
  - Adaptive procedures (e.g. linesearch)
- **Prefer adaptive procedures (no tuning, fits to local curvature)**

Derivative-Free Optimisation

For black-box, computationally expensive and/or noisy functions, we cannot assume access to gradient information.

Alternative: derivative-free optimisation (DFO)

Derivative-Free Optimisation

For black-box, computationally expensive and/or noisy functions, we cannot assume access to gradient information.

Alternative: derivative-free optimisation (DFO)

- Assume can evaluate $f(\boldsymbol{x})$ but not $\nabla f(\boldsymbol{x})$ (but still assume f is differentiable)

Derivative-Free Optimisation

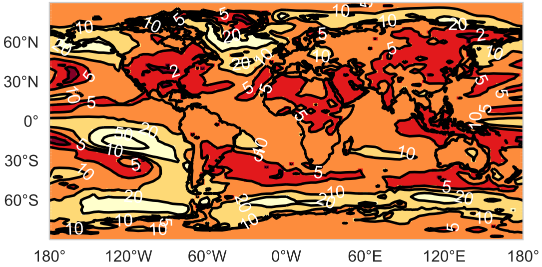
For black-box, computationally expensive and/or noisy functions, we cannot assume access to gradient information.

Alternative: derivative-free optimisation (DFO)

- Assume can evaluate $f(\boldsymbol{x})$ but not $\nabla f(\boldsymbol{x})$ (but still assume f is differentiable)
- Several approaches: Nelder-Mead, genetic algorithms, Bayesian optimisation, ...
- Seek local minimiser (actually, approximate stationary point: $\|\nabla f(\boldsymbol{x})\|_2 \leq \epsilon$)
- Focus on efficient & adaptive methods

Application 1: Climate Modelling [Tett et al., 2022]

- Parameter calibration for global climate models (least squares minimisation)
- One model run = simulate global climate for 5 years = expensive
- Very complicated, chaotic physics = black-box & noisy



Application 2: Adversarial Example Generation [Alzantot et al., 2019]

- Find perturbations of neural network inputs which are misclassified (min. probability of correct label/max. probability of desired incorrect label)
- Neural network structure assumed to be unknown = black-box
- Want to test very few examples ≈ expensive
- Useful for copyright protection of artists' work against generative AI [Shan et al., 2023]

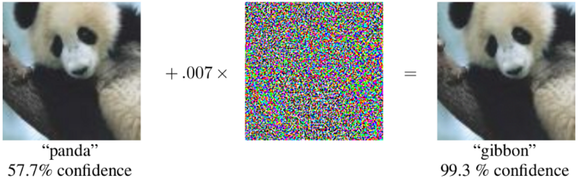


Image from [Goodfellow et al., 2015]

Application 3: Fine-Tuning Large Language Models [Malladi et al., 2023]

- Take pre-trained LLM, tweak parameters to be better at a specific task
  - e.g. Sentiment analysis: "[input text]. It was..." (good or bad?)
- Very large models = backpropagation expensive & distributed
- DFO method (MeZO) uses 12x less memory than gradient-based methods (FT) but with comparable performance

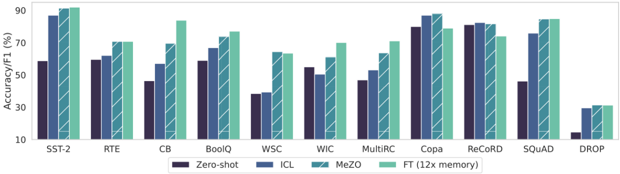


Image from [Malladi et al., 2023]

Direct Search

Method 1: Direct Search (simple & easily generalised)

Method 1: Direct Search (simple & easily generalised)

- Given $\boldsymbol{x}_k \in \mathbb{R}^n$ and $\Delta_k > 0$, choose a set $\mathcal{D}_k \subset \mathbb{R}^n$ of m vectors
- If there exists $\boldsymbol{d}_k \in \mathcal{D}_k$ with $f(\boldsymbol{x}_k + \Delta_k \boldsymbol{d}_k) < f(\boldsymbol{x}_k) - \frac{1}{2}\Delta_k^2 \|\boldsymbol{d}_k\|_2^2$
  - Set $\boldsymbol{x}_{k+1} = \boldsymbol{x}_k + \Delta_k \boldsymbol{d}_k$ and increase Δ_k
  - Otherwise, set $\boldsymbol{x}_{k+1} = \boldsymbol{x}_k$ and decrease Δ_k

Direct Search

Method 1: Direct Search (simple & easily generalised)

- Given $\boldsymbol{x}_k \in \mathbb{R}^n$ and $\Delta_k > 0$, choose a set $\mathcal{D}_k \subset \mathbb{R}^n$ of m vectors
- If there exists $\boldsymbol{d}_k \in \mathcal{D}_k$ with $f(\boldsymbol{x}_k + \Delta_k \boldsymbol{d}_k) < f(\boldsymbol{x}_k) - \frac{1}{2}\Delta_k^2 \|\boldsymbol{d}_k\|_2^2$
  - Set $\boldsymbol{x}_{k+1} = \boldsymbol{x}_k + \Delta_k \boldsymbol{d}_k$ and increase Δ_k
  - Otherwise, set $\boldsymbol{x}_{k+1} = \boldsymbol{x}_k$ and decrease Δ_k

For convergence, need $\mathcal{D}_k$ to be κ-descent:

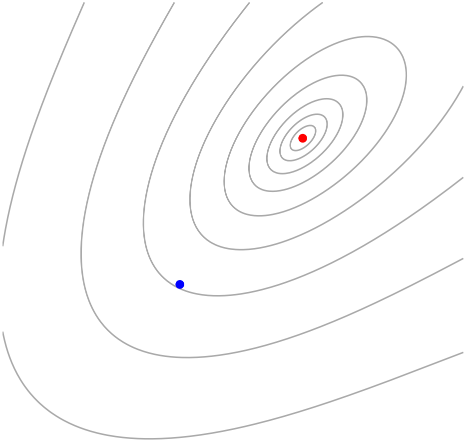
$$\max_{\boldsymbol{d} \in \mathcal{D}_k} \frac{-\boldsymbol{d}^T \nabla f(\boldsymbol{x}_k)}{\|\boldsymbol{d}\|_2 \cdot \|\nabla f(\boldsymbol{x}_k)\|_2} \geq \kappa \in (0, 1]$$

i.e. there is a vector $\boldsymbol{d}$ making an acute angle with $-\nabla f(\boldsymbol{x}_k)$.

Examples: $\{\pm \boldsymbol{e}_1, \ldots, \pm \boldsymbol{e}_n\}$ with $\kappa = 1/\sqrt{n}$ or $\{\boldsymbol{e}_1, \ldots, \boldsymbol{e}_n, -\boldsymbol{e}\}$ with $\kappa \sim 1/n$.

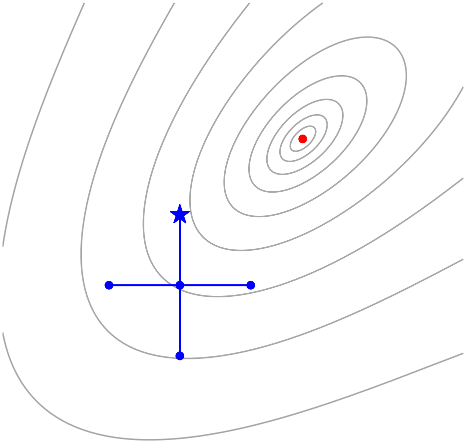
[Kolda, Lewis & Torczon, 2003; Conn, Scheinberg & Vicente, 2009]

Example: Direct Search



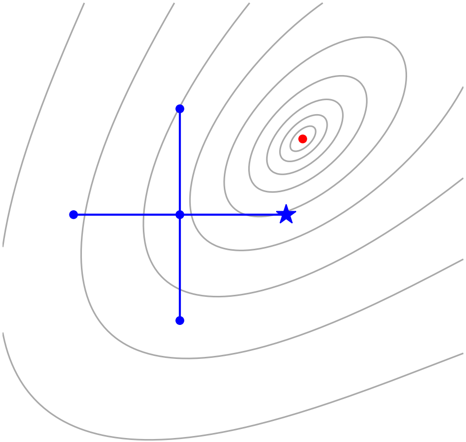
Modified from [Kolda, Lewis & Torczon, 2003]

Example: Direct Search



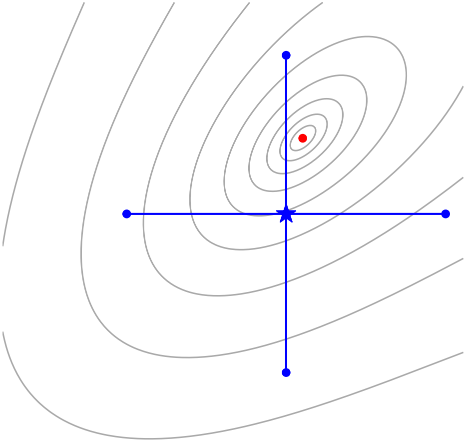
Modified from [Kolda, Lewis & Torczon, 2003]

Example: Direct Search



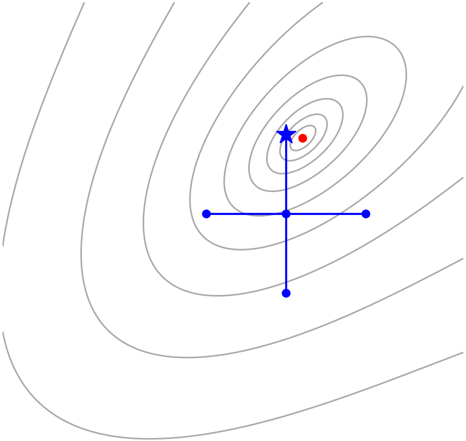
Modified from [Kolda, Lewis & Torczon, 2003]

Example: Direct Search



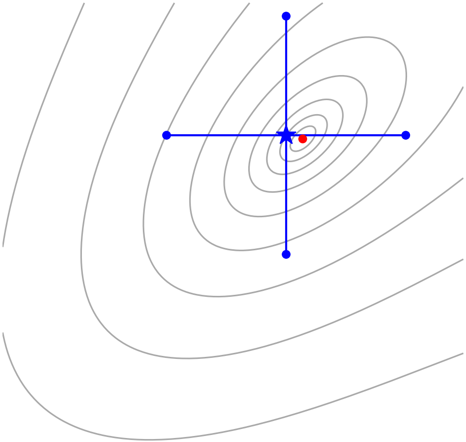
Modified from [Kolda, Lewis & Torczon, 2003]

Example: Direct Search



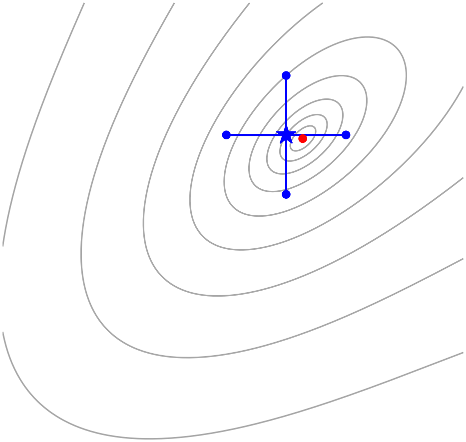
Modified from [Kolda, Lewis & Torczon, 2003]

Example: Direct Search



Modified from [Kolda, Lewis & Torczon, 2003]

Example: Direct Search



Modified from [Kolda, Lewis & Torczon, 2003]

Method 2: Model-Based Optimisation (c.f. Bayesian optimisation)

- Build a Taylor series-like model

$$f(\mathbf{x}_k + \mathbf{s}) \approx m_k(\mathbf{s}) = f(\mathbf{x}_k) + \mathbf{g}_k^T \mathbf{s} + \frac{1}{2}\mathbf{s}^T H_k \mathbf{s}$$

- Get step by minimising model in a neighbourhood

$$\mathbf{s}_k = \arg\min_{\mathbf{s}\in\mathbb{R}^n} m_k(\mathbf{s}) \qquad \text{subject to } \|\mathbf{s}\|_2 \leq \Delta_k$$

$\implies$ *'trust region' subproblem – specialised algorithms exist*

- Accept/reject step and adjust Δ_k based on quality of new point $f(\mathbf{x}_k + \mathbf{s}_k)$

$$\mathbf{x}_{k+1} = \begin{cases} \mathbf{x}_k + \mathbf{s}_k, & \text{if sufficient decrease,} \qquad \longleftarrow \text{(maybe increase } \Delta_k) \\ \mathbf{x}_k, & \text{otherwise.} \qquad \longleftarrow \text{(decrease } \Delta_k) \end{cases}$$

Model-Based Optimisation

- Build a Taylor series-like model

$$f(\boldsymbol{x}_k + \boldsymbol{s}) \approx m_k(\boldsymbol{s}) = f(\boldsymbol{x}_k) + \boldsymbol{g}_k{}^T \boldsymbol{s} + \frac{1}{2}\boldsymbol{s}^T H_k \boldsymbol{s}$$

  and find $\boldsymbol{g}_k$ and H_k without using derivatives
- How?

Model-Based Optimisation

- Build a Taylor series-like model

$$f(\mathbf{x}_k + \mathbf{s}) \approx m_k(\mathbf{s}) = f(\mathbf{x}_k) + \mathbf{g}_k{}^T \mathbf{s} + \frac{1}{2}\mathbf{s}^T H_k \mathbf{s}$$

  and find $\mathbf{g}_k$ and H_k without using derivatives
- How? Interpolate f over a set of points — find $\mathbf{g}_k$, H_k such that

$$m_k(\mathbf{y} - \mathbf{x}_k) = f(\mathbf{y}), \qquad \forall \mathbf{y} \in \mathcal{Y}$$

Model-Based Optimisation

- Build a Taylor series-like model

$$f(\boldsymbol{x}_k + \boldsymbol{s}) \approx m_k(\boldsymbol{s}) = f(\boldsymbol{x}_k) + \boldsymbol{g}_k{}^T \boldsymbol{s} + \frac{1}{2}\boldsymbol{s}^T H_k \boldsymbol{s}$$

  and find $\boldsymbol{g}_k$ and H_k without using derivatives

- How? Interpolate f over a set of points — find $\boldsymbol{g}_k$, H_k such that

$$m_k(\boldsymbol{y} - \boldsymbol{x}_k) = f(\boldsymbol{y}), \qquad \forall \boldsymbol{y} \in \mathcal{Y}$$

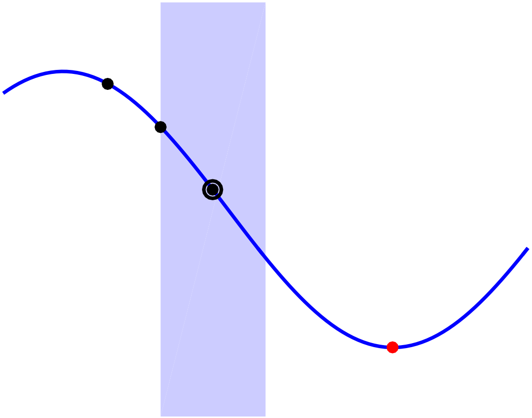
For convergence, need m_k to be fully linear:

$$|f(\boldsymbol{x}_k + \boldsymbol{s}) - m_k(\boldsymbol{s})| \leq \mathcal{O}(\Delta_k^2) \qquad \text{and} \qquad \|\nabla f(\boldsymbol{x}_k + \boldsymbol{s}) - \nabla m_k(\boldsymbol{s})\|_2 \leq \mathcal{O}(\Delta_k)$$

Achievable if points in $\mathcal{Y}$ are well-spaced (in a specific sense).

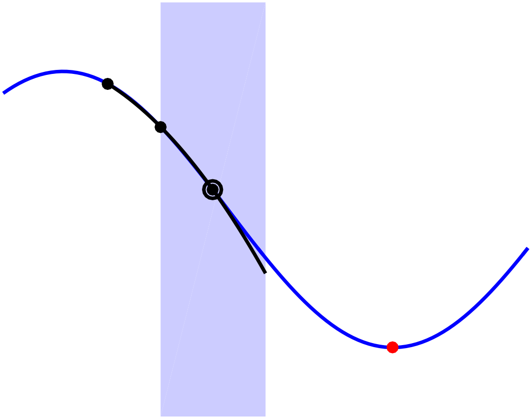
[Powell, 2003; Conn, Scheinberg & Vicente, 2009]

Example: Model-Based DFO



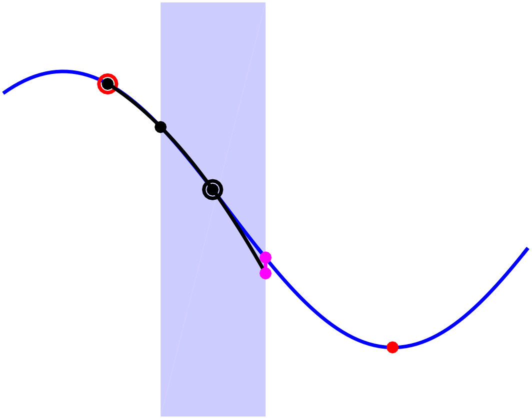
1. Choose interpolation set

Example: Model-Based DFO



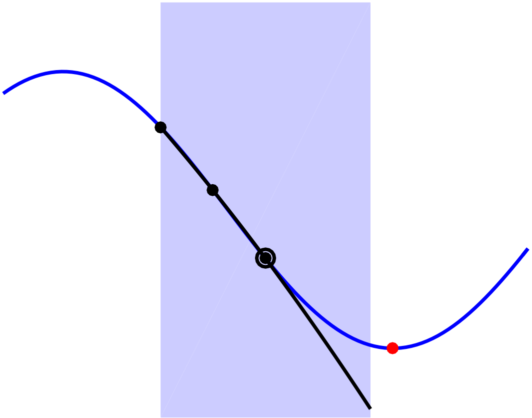
2. Interpolate & minimise...

Example: Model-Based DFO



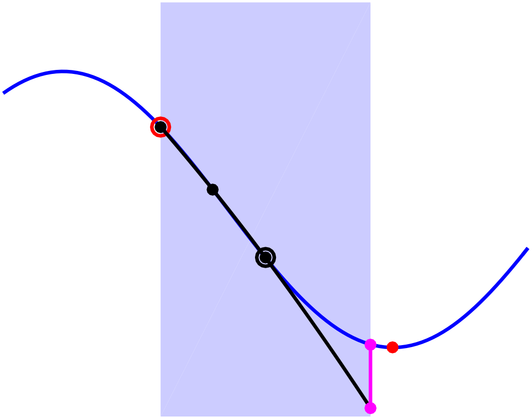
3. Add new point to interpolation set (replace a bad point)

Example: Model-Based DFO



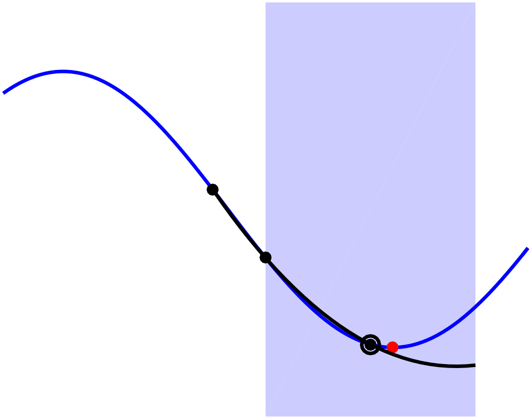
4. Repeat with new interpolation set & model

Example: Model-Based DFO



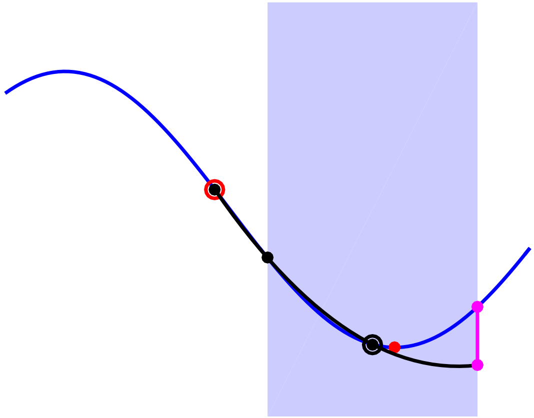
4. Repeat with new interpolation set & model

Example: Model-Based DFO



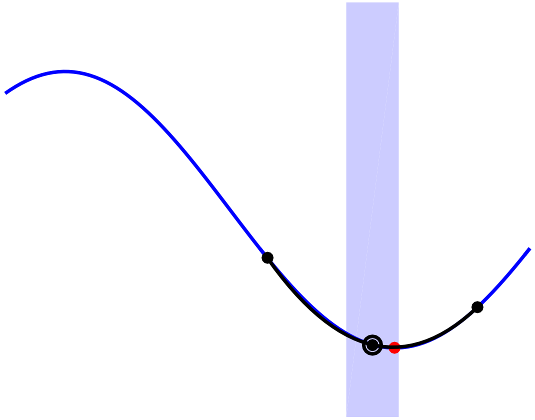
4. Repeat with new interpolation set & model

Example: Model-Based DFO



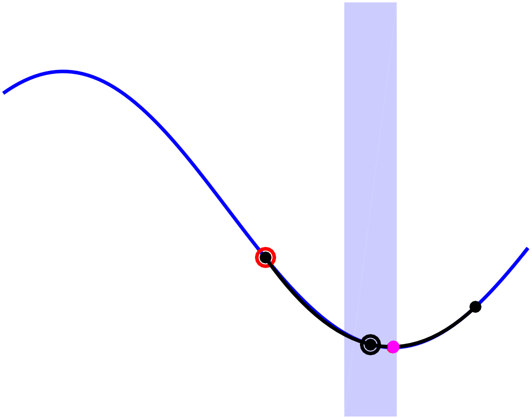
4. Repeat with new interpolation set & model

Example: Model-Based DFO



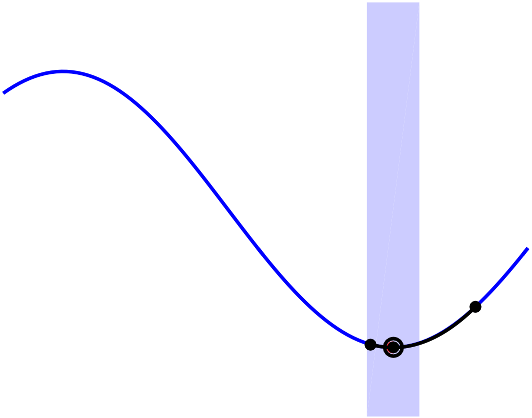
4. Repeat with new interpolation set & model

Example: Model-Based DFO



4. Repeat with new interpolation set & model

Example: Model-Based DFO



4. Repeat with new interpolation set & model

Complexity Theory

Analyse methods using **worst-case complexity**: how long before $\|\nabla f(\boldsymbol{x}_k)\|_2 \leq \epsilon$?

Metric	Deriv-based	Model-based	Direct search
Iterations	$\mathcal{O}(\epsilon^{-2})$	$\mathcal{O}(n^2\epsilon^{-2})$	$\mathcal{O}(n\epsilon^{-2})$
Evaluations	$\approx \mathcal{O}(n\epsilon^{-2})$	$\mathcal{O}(n^3\epsilon^{-2})$	$\mathcal{O}(n^2\epsilon^{-2})$

[Cartis, Gould & Toint, 2010; Garmanjani, Júdice & Vicente, 2016; Vicente, 2013]

- Same ϵ dependency as derivative-based, but scales badly with problem dimension n
- Model-based methods also have substantial linear algebra work for interpolation and geometry management: at least $\mathcal{O}(n^3)$ flops per iteration

Challenge

How can DFO methods be made scalable?

Complexity Theory

Analyse methods using **worst-case complexity**: how long before $\|\nabla f(\mathbf{x}_k)\|_2 \leq \epsilon$?

Metric	Deriv-based	Model-based	Direct search
Iterations	$\mathcal{O}(\epsilon^{-2})$	$\mathcal{O}(n^2\epsilon^{-2})$	$\mathcal{O}(n\epsilon^{-2})$
Evaluations	$\approx \mathcal{O}(n\epsilon^{-2})$	~~$\mathcal{O}(n^3\epsilon^{-2})$~~ $\mathcal{O}(n^2\epsilon^{-2})$	~~$\mathcal{O}(n^2\epsilon^{-2})$~~ $\mathcal{O}(n\epsilon^{-2})$

[Cartis, Gould & Toint, 2010; Garmanjani, Júdice & Vicente, 2016; Vicente, 2013]

- Same ϵ dependency as derivative-based, but ~~scales badly with problem dimension n~~
- Model-based methods also have substantial linear algebra work for interpolation and geometry management: at least ~~$\mathcal{O}(n^3)$~~ $\mathcal{O}(n)$ flops per iteration

Challenge

How can DFO methods be made scalable?

Outline

1. Introduction to derivative-free optimisation (DFO)

2. **Subspace DFO methods**

3. Average-case analysis

Randomised methods

Challenge

How can DFO methods be made scalable?

The machine learning community often uses randomised finite differencing ('gradient sampling')

$$\nabla f(\boldsymbol{x}) \approx \left[\frac{f(\boldsymbol{x}+h\boldsymbol{v})-f(\boldsymbol{x})}{h}\right]\boldsymbol{v},$$

for random $\boldsymbol{v}$ (e.g. standard Gaussian). [Ghadimi & Lan, 2013; Nesterov & Spokoiny, 2017]

Randomised methods

Challenge

How can DFO methods be made scalable?

The machine learning community often uses randomised finite differencing ('gradient sampling')

$$\nabla f(\boldsymbol{x}) \approx \left[\frac{f(\boldsymbol{x}+h\boldsymbol{v}) - f(\boldsymbol{x})}{h}\right]\boldsymbol{v},$$

for random $\boldsymbol{v}$ (e.g. standard Gaussian). [Ghadimi & Lan, 2013; Nesterov & Spokoiny, 2017]

- Better complexity, but still need expensive hyperparameter tuning
- More structure in sampling (e.g. fully linear requirements) gives better gradient estimates [Berahas et al., 2022]

Randomised methods

Challenge

How can DFO methods be made scalable?

Randomisation is still a promising approach:

- Make search directions κ-descent with probability < 1 [Gratton et al., 2015]
- Make model fully linear with probability < 1 [Gratton et al., 2017]

Problem: Improves complexity for direct search, but not for model-based!

Randomised methods

Challenge

How can DFO methods be made scalable?

Randomisation is still a promising approach:

- Make search directions κ-descent with probability < 1 [Gratton et al., 2015]
- Make model fully linear with probability < 1 [Gratton et al., 2017]

Problem: Improves complexity for direct search, but not for model-based!

Why? Direct search formulation effectively allows dimensionality reduction (sample $\ll n$ directions).

Goal

Use dimensionality reduction techniques suitable for both classes.

Lemma (Johnson-Lindenstrauss, 1984)

Suppose $\boldsymbol{x}_1, \dots, \boldsymbol{x}_N \in \mathbb{R}^d$ *and* $\epsilon \in (0,1)$. *Let* $A \in \mathbb{R}^{p \times d}$ *be a matrix with i.i.d.* $\mathcal{N}(0, p^{-2})$ *entries and* $p \sim \log(N)/\epsilon$. *Then with high probability,*

$$(1-\epsilon)\|\boldsymbol{x}_i - \boldsymbol{x}_j\|_2 \leq \|A\boldsymbol{x}_i - A\boldsymbol{x}_j\|_2 \leq (1+\epsilon)\|\boldsymbol{x}_i - \boldsymbol{x}_j\|_2, \qquad \forall i,j = 1, \dots, N.$$

Randomisation for Dimensionality Reduction

Lemma (Johnson-Lindenstrauss, 1984)

Suppose $\mathbf{x}_1, \ldots, \mathbf{x}_N \in \mathbb{R}^d$ *and* $\epsilon \in (0,1)$. *Let* $A \in \mathbb{R}^{p\times d}$ *be a matrix with i.i.d.* $\mathcal{N}(0, p^{-2})$ *entries and* $p \sim \log(N)/\epsilon$. *Then with high probability,*

$$(1-\epsilon)\|\mathbf{x}_i - \mathbf{x}_j\|_2 \le \|A\mathbf{x}_i - A\mathbf{x}_j\|_2 \le (1+\epsilon)\|\mathbf{x}_i - \mathbf{x}_j\|_2, \qquad \forall i,j = 1,\ldots,N.$$

- Random projections approximately preserve distances (& inner products, norms, ...)
- Reduced dimension p depends only on # of points N, not the ambient dimension d!
- Other random constructions satisfy J-L Lemma (Haar subsampling, hashing, ...)

Subspace methods

We use a subspace method: only search in low-dimensional subspaces of $\mathbb{R}^n$

Subspace methods

We use a subspace method: only search in low-dimensional subspaces of $\mathbb{R}^n$

Subspace framework:

- Generate subspace of dimension $p \ll n$ given by $\text{col}(P_k)$ for random $P_k \in \mathbb{R}^{n \times p}$
- Direct search: choose $\mathcal{D}_k \subset \mathbb{R}^p$ which is κ-descent for $P_k^T \nabla f(\boldsymbol{x}_k) \in \mathbb{R}^p$
- Model-based: build a low-dimensional model $\hat{m}_k(\hat{\boldsymbol{s}})$ which is fully linear for $\hat{f}(\hat{\boldsymbol{s}}) := f(\boldsymbol{x}_k + P_k \hat{\boldsymbol{s}}) : \mathbb{R}^p \to \mathbb{R}$

Fewer interpolation/sample points needed, cheap linear algebra (everything in $\mathbb{R}^p$)

Subspace methods — Subspace Quality

Choice of subspace: we need to make sure we search in 'good' subspaces (where there is potential to decrease f sufficiently).

The subspace at iteration k is well-aligned if

$$\|P_k^T \nabla f(\boldsymbol{x}_k)\|_2 \geq \alpha \|\nabla f(\boldsymbol{x}_k)\|_2, \qquad \text{for some } \alpha > 0.$$

i.e. if there is still work to do, then we know this by only inspecting f in the subspace.

Subspace methods — Subspace Quality

Choice of subspace: we need to make sure we search in 'good' subspaces (where there is potential to decrease f sufficiently).

The subspace at iteration k is well-aligned if

$$\|P_k^T \nabla f(\boldsymbol{x}_k)\|_2 \geq \alpha \|\nabla f(\boldsymbol{x}_k)\|_2, \qquad \text{for some } \alpha > 0.$$

i.e. if there is still work to do, then we know this by only inspecting f in the subspace.

Key Assumption

The subspace P_k is well-aligned with probability $1 - \delta$.

Using J-L lemma, choose $p \sim (1-\alpha)^{-2} |\log \delta| = \mathcal{O}(1)$ independent of n.

Data oblivious: don't need to know $\nabla f(\boldsymbol{x}_k)$ when generating P_k.

Subspace DFO — Complexity

Theorem (Cartis & LR, 2023; LR & Royer, 2023)

If f is sufficiently smooth and bounded below and ϵ sufficiently small, then

$$\mathbb{P}\left[K_\epsilon \leq C(p, \alpha, \delta)\epsilon^{-2}\right] \geq 1 - e^{-c(p,\alpha,\delta)\epsilon^{-2}},$$

where K_ϵ is the first iteration with $\|\nabla f(\boldsymbol{x}_k)\|_2 \leq \epsilon$.

- Implies $\mathbb{E}[K_\epsilon] = \mathcal{O}(\epsilon^{-2})$ and $\inf_k \|\nabla f(\boldsymbol{x}_k)\|_2 = 0$ almost surely
- $\mathcal{O}(p)$ evaluations per iteration, so same bounds for evaluation complexity

Subspace methods — Complexity

Standard methods:

Metric	Deriv-based	Model-based	Direct search	Rand. FD
Iterations	$\mathcal{O}(\epsilon^{-2})$	$\mathcal{O}(n^2\epsilon^{-2})$	$\mathcal{O}(n\epsilon^{-2})$	$\mathcal{O}(n\epsilon^{-2})$
Evaluations	$\approx \mathcal{O}(n\epsilon^{-2})$	$\mathcal{O}(n^3\epsilon^{-2})$	$\mathcal{O}(n^2\epsilon^{-2})$	$\mathcal{O}(n\epsilon^{-2})$

Model-based methods have $\mathcal{O}(n^3)$ linear algebra work per iteration.

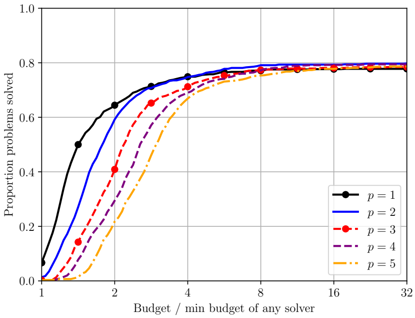
Using random subspaces:

Metric	Deriv-based	Model-based	Direct search	Rand. FD
Iterations	$\mathcal{O}(\epsilon^{-2})$	$\mathcal{O}(n^2\epsilon^{-2})$	$\mathcal{O}(n\epsilon^{-2})$	$\mathcal{O}(n\epsilon^{-2})$
Evaluations	$\approx \mathcal{O}(n\epsilon^{-2})$	$\mathcal{O}(n^2\epsilon^{-2})$	$\mathcal{O}(n\epsilon^{-2})$	$\mathcal{O}(n\epsilon^{-2})$

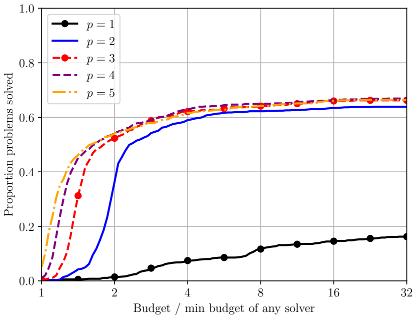
Model-based methods have $\mathcal{O}(n)$ linear algebra work per iteration.

Example Results

Example results for different subspace dimensions p:



Direct Search

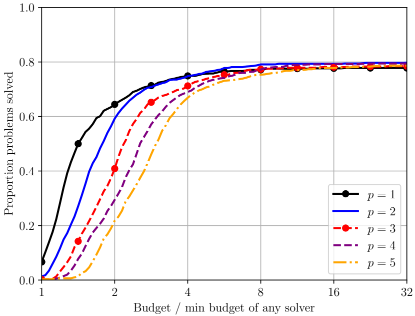


Model-Based

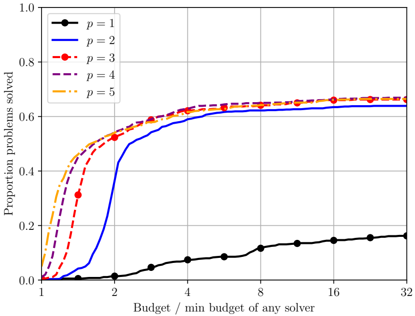
Fraction of test problems solved vs. # evaluations of f — higher is better.

Example Results

Example results for different subspace dimensions p:



Direct Search



Model-Based

Theory says $p = \mathcal{O}(1)$ works, numerics say take $p \to \sim 1$. Why might this be true?

Outline

1. Introduction to derivative-free optimisation (DFO)
2. Subspace DFO methods
3. **Average-case analysis**

Average-Case Analysis

Almost all analysis of optimisation algorithms is worst-case: e.g. "for all objectives f in a given class, get $\|\nabla f(\boldsymbol{x}_k)\|_2 \leq \epsilon$ after at most $k = \mathcal{O}(\epsilon^{-2})$ iterations".

Does this capture realistic behaviour?

Average-Case Analysis

Almost all analysis of optimisation algorithms is worst-case: e.g. "for all objectives f in a given class, get $\|\nabla f(\boldsymbol{x}_k)\|_2 \le \epsilon$ after at most $k = \mathcal{O}(\epsilon^{-2})$ iterations".

Does this capture realistic behaviour?

- Not for linear programming! Simplex method takes exponentially many iterations (worst-case) but on average is polynomial time [Spielman & Teng, 2004]
- Gradient descent-type methods designed for (convex) average-case Hessian spectra can outperform "worst-case optimal" methods [Pedregosa & Scieur, 2020]
- For nonconvex optimisation, can do worst-case analysis in different regions of the domain separately [Curtis & Robinson, 2021]

New here: average-case analysis for nonconvex optimisation algorithms.

What is a tractable model to analyse these algorithms?

Average-Case Analysis

What is a tractable model to analyse these algorithms?

- Pick random linear function $f(\boldsymbol{x}) = \boldsymbol{v}^T \boldsymbol{x}$
- At $\boldsymbol{x}_k$, pick a random p-dimensional subspace
- Do 1 iteration of subspace method in dimension p
  - Direct search with $\mathcal{D}_k = \{\pm \boldsymbol{e}_1, \ldots, \pm \boldsymbol{e}_p\}$ or model-based with linear interpolation
- Look at expected decrease as function of relevant dimensions

$$\mathbb{E}(p, n) := \mathbb{E}[f(\boldsymbol{x}_k) - f(\boldsymbol{x}_{k+1})]$$

with expectation over uniformly distributed objective functions (unit vectors $\boldsymbol{v}$) and subspaces (Stiefel manifold).

Average-Case Analysis

What is a tractable model to analyse these algorithms?

- Pick random linear function $f(\boldsymbol{x}) = \boldsymbol{v}^T \boldsymbol{x}$
- At $\boldsymbol{x}_k$, pick a random p-dimensional subspace
- Do 1 iteration of subspace method in dimension p
  - Direct search with $\mathcal{D}_k = \{\pm \boldsymbol{e}_1, \ldots, \pm \boldsymbol{e}_p\}$ or model-based with linear interpolation
- Look at expected decrease as function of relevant dimensions

$$\mathbb{E}(p, n) := \mathbb{E}[f(\boldsymbol{x}_k) - f(\boldsymbol{x}_{k+1})]$$

with expectation over uniformly distributed objective functions (unit vectors $\boldsymbol{v}$) and subspaces (Stiefel manifold).

Assumes f is linear, or $\Delta_k \ll 1$, i.e. close to a solution.

Average-Case Analysis: Direct Search

Calculating expected decrease leads to an interesting problem:

Lemma

For direct search, $\mathbb{E}(p, n) = \mathbb{E}_{\boldsymbol{g}\sim\mathbb{S}^{n-1}}[\max(|g_1|, \ldots, |g_p|)]$

i.e. for a randomly distributed unit vector $\boldsymbol{g} \in \mathbb{R}^n$, $\|\boldsymbol{g}\|_2 = 1$, what is the expected ∞-norm of its first p coordinates?

Average-Case Analysis: Direct Search

Calculating expected decrease leads to an interesting problem:

Lemma

For direct search, $\mathbb{E}(p,n) = \mathbb{E}_{\boldsymbol{g}\sim\mathbb{S}^{n-1}}[\max(|g_1|,\ldots,|g_p|)]$

i.e. for a randomly distributed unit vector $\boldsymbol{g}\in\mathbb{R}^n$, $\|\boldsymbol{g}\|_2 = 1$, what is the expected ∞-norm of its first p coordinates?

Theorem (Hare, LR & Royer, 2023)

For direct search,

$$\mathbb{E}(p,n) = \frac{p2^{p-1}}{\pi^{p/2}} \cdot \frac{\Gamma\left(\frac{n}{2}\right)\Gamma\left(\frac{p+1}{2}\right)}{\Gamma\left(\frac{n+1}{2}\right)} \cdot \mathcal{I}(p)$$

where $\mathcal{I}(p)$ *is a (nasty)* $(p-1)$*-dimensional integral.*

Nasty Integral

$$\mathcal{I}(p) = \int_R \left[\prod_{j=1}^{p-1} \sin^j(\varphi_j)\right] d\varphi_{p-1} \cdots d\varphi_1$$

where

$$R = \left\{(\varphi_1, \ldots, \varphi_{p-1}) \in \left[\frac{\pi}{4}, \frac{\pi}{2}\right] \times \prod_{j=2}^{p-1} \left[\arctan\left(\prod_{k=1}^{j-1} \frac{1}{\sin(\varphi_k)}\right), \frac{\pi}{2}\right]\right\}$$

Nasty Integral

$$\mathcal{I}(p) = \int_R \left[\prod_{j=1}^{p-1} \sin^j(\varphi_j)\right] d\varphi_{p-1}\cdots d\varphi_1$$

where

$$R = \left\{(\varphi_1, \ldots, \varphi_{p-1}) \in \left[\frac{\pi}{4}, \frac{\pi}{2}\right] \times \prod_{j=2}^{p-1}\left[\arctan\left(\prod_{k=1}^{j-1}\frac{1}{\sin(\varphi_k)}\right), \frac{\pi}{2}\right]\right\}$$

p	$\mathcal{I}(p)$	Approx.
1	1	1.0000
2	$1/\sqrt{2}$	0.7071
3	$\left(4\arctan(\sqrt{2}) + \arctan(460\sqrt{2}/329)\right)/(8\sqrt{2})$	0.4352
4	$\arctan(1/(2\sqrt{2}))/\sqrt{2}$	0.2403

Average-Case Analysis: Direct Search

Although $\mathcal{I}(p)$ is nasty, we can still get bounds on it and then look at "expected decrease per objective evaluation".

Average-Case Analysis: Direct Search

Although $\mathcal{I}(p)$ is nasty, we can still get bounds on it and then look at "expected decrease per objective evaluation".

Theorem (Hare, LR & Royer, 2023)

For any n, the expected decrease per objective evaluation for direct search, $\mathbb{E}(p, n)/(2p)$, is strictly decreasing in p for $p = 1, \ldots, n$.

Average-Case Analysis: Direct Search

Although $\mathcal{I}(p)$ is nasty, we can still get bounds on it and then look at "expected decrease per objective evaluation".

Theorem (Hare, LR & Royer, 2023)

For any n, the expected decrease per objective evaluation for direct search, $\mathbb{E}(p, n)/(2p)$, is strictly decreasing in p for $p = 1, \ldots, n$.

So, the smallest subspace dimension $p = 1$ gives the best 'bang for your buck'.

Average-Case Analysis: Model-Based

For model-based methods, look at expected 2-norm of first p components of random unit vector (much nicer than ∞-norm) to get a similar result:

$$\mathbb{E}(p,n) = \mathbb{E}_{\boldsymbol{g}\sim\mathbb{S}^{n-1}}\left[\sqrt{g_1^2+\cdots+g_p^2}\right] = \frac{\Gamma\left(\frac{n}{2}\right)\cdot\Gamma\left(\frac{p+1}{2}\right)}{\Gamma\left(\frac{n+1}{2}\right)\cdot\Gamma\left(\frac{p}{2}\right)} \qquad \approx \frac{\sqrt{p}}{\sqrt{n}} \text{ for } p, n \text{ large}$$

Average-Case Analysis: Model-Based

For model-based methods, look at expected 2-norm of first p components of random unit vector (much nicer than ∞-norm) to get a similar result:

$$\mathbb{E}(p,n) = \mathbb{E}_{\boldsymbol{g}\sim\mathbb{S}^{n-1}}\left[\sqrt{g_1^2+\cdots+g_p^2}\right] = \frac{\Gamma\left(\frac{n}{2}\right)\cdot\Gamma\left(\frac{p+1}{2}\right)}{\Gamma\left(\frac{n+1}{2}\right)\cdot\Gamma\left(\frac{p}{2}\right)} \qquad \approx \frac{\sqrt{p}}{\sqrt{n}} \text{ for } p,n \text{ large}$$

Theorem (Hare, LR & Royer, 2023)

For any n, the expected decrease per objective evaluation, $\mathbb{E}(p,n)/(p+1)$, satisfies

$$\frac{\mathbb{E}(2,n)}{3} > \left[\frac{\mathbb{E}(1,n)}{2} = \frac{\mathbb{E}(3,n)}{4}\right] > \frac{\mathbb{E}(4,n)}{5} > \cdots > \frac{\mathbb{E}(n,n)}{n+1}$$

Average-Case Analysis: Model-Based

For model-based methods, look at expected 2-norm of first p components of random unit vector (much nicer than ∞-norm) to get a similar result:

$$\mathbb{E}(p,n) = \mathbb{E}_{\boldsymbol{g}\sim\mathbb{S}^{n-1}}\left[\sqrt{g_1^2+\cdots+g_p^2}\right] = \frac{\Gamma\left(\frac{n}{2}\right)\cdot\Gamma\left(\frac{p+1}{2}\right)}{\Gamma\left(\frac{n+1}{2}\right)\cdot\Gamma\left(\frac{p}{2}\right)} \approx \frac{\sqrt{p}}{\sqrt{n}} \text{ for } p,n \text{ large}$$

Theorem (Hare, LR & Royer, 2023)

For any n, the expected decrease per objective evaluation, $\mathbb{E}(p,n)/(p+1)$, satisfies

$$\frac{\mathbb{E}(2,n)}{3} > \left[\frac{\mathbb{E}(1,n)}{2} = \frac{\mathbb{E}(3,n)}{4}\right] > \frac{\mathbb{E}(4,n)}{5} > \cdots > \frac{\mathbb{E}(n,n)}{n+1}$$

So $\mathbb{E}(p,n)/(p+1)$ is strictly decreasing in p for $p \geq 2$, not $p \geq 1$.

Conclusions & Future Work

Conclusions

- DFO useful for optimising complex/expensive functions
- Randomised projections can be effective for dimensionality reduction
- Large-scale DFO is possible using random subspaces

Conclusions & Future Work

Conclusions

- DFO useful for optimising complex/expensive functions
- Randomised projections can be effective for dimensionality reduction
- Large-scale DFO is possible using random subspaces

Future Work

- Second-order worst-case complexity analysis
- Efficient implementation of subspace quadratic models (model-based)
- Average-case analysis for quadratic objectives
- Impact of noisy objective evaluations
- Impact of low effective dimensionality
- Constrained problems?

References i

M. Alzantot, Y. Sharma, S. Chakraborty, H. Zhang, C.-J. Hsieh, and M. B. Srivastava, *GenAttack: Practical black-box attacks with gradient-free optimization*, in Proceedings of the Genetic and Evolutionary Computation Conference, Prague, Czech Republic, 2019, ACM, pp. 1111–1119.

A. S. Berahas, L. Cao, K. Choromanski, and K. Scheinberg, *A theoretical and empirical comparison of gradient approximations in derivative-free optimization*, Foundations of Computational Mathematics, 22 (2022), pp. 507–560.

C. Cartis, N. I. M. Gould, and P. L. Toint, *On the complexity of steepest descent, Newton's and regularized Newton's methods for nonconvex unconstrained optimization problems*, SIAM Journal on Optimization, 20 (2010), pp. 2833–2852.

C. Cartis and L. Roberts, *Scalable subspace methods for derivative-free nonlinear least-squares optimization*, Mathematical Programming, 199 (2023), pp. 461—524.

A. R. Conn, K. Scheinberg, and L. N. Vicente, *Introduction to Derivative-Free Optimization*, vol. 8 of MPS-SIAM Series on Optimization, MPS/SIAM, Philadelphia, 2009.

F. E. Curtis and D. P. Robinson, *Regional complexity analysis of algorithms for nonconvex smooth optimization*, Mathematical Programming, 187 (2021), pp. 579–615.

R. Garmanjani, D. Júdice, and L. N. Vicente, *Trust-region methods without using derivatives: Worst case complexity and the nonsmooth case*, SIAM Journal on Optimization, 26 (2016), pp. 1987–2011.

S. Ghadimi and G. Lan, *Stochastic first- and zeroth-order methods for nonconvex stochastic programming*, SIAM Journal on Optimization, 23 (2013), pp. 2341–2368.

I. J. Goodfellow, J. Shlens, and C. Szegedy, *Explaining and harnessing adversarial examples*, in 3rd International Conference on Learning Representations ICLR, San Diego, 2015.

S. Gratton, C. W. Royer, L. N. Vicente, and Z. Zhang, *Direct search based on probabilistic descent*, SIAM Journal on Optimization, 25 (2015), pp. 1515–1541.

S. Gratton, C. W. Royer, L. N. Vicente, and Z. Zhang, *Complexity and global rates of trust-region methods based on probabilistic models*, IMA Journal of Numerical Analysis, 38 (2017), pp. 1579–1597.

W. Hare, L. Roberts, and C. W. Royer, *Expected decrease for derivative-free algorithms using random subspaces*, Mathematics of Computation, (2024).

References iii

W. B. Johnson and J. Lindenstrauss, *Extensions of Lipschitz mappings into a Hilbert space*, in Contemporary Mathematics, R. Beals, A. Beck, A. Bellow, and A. Hajian, eds., vol. 26, American Mathematical Society, Providence, Rhode Island, 1984, pp. 189–206.

T. G. Kolda, R. M. Lewis, and V. Torczon, *Optimization by direct search: New perspectives on some classical and modern methods*, SIAM Review, 45 (2003), pp. 385–482.

S. Malladi, T. Gao, E. Nichani, A. Damian, J. D. Lee, D. Chen, and S. Arora, *Fine-tuning language models with just forward passes*, arXiv preprint arXiv:2305.17333, (2023).

Y. Nesterov and V. Spokoiny, *Random gradient-free minimization of convex functions*, Foundations of Computational Mathematics, 17 (2017), pp. 527–566.

F. Pedregosa and D. Scieur, *Average-case acceleration through spectral density estimation*, Proceedings of the 37th International Conference on Machine Learning, (2020).

M. J. D. Powell, *On trust region methods for unconstrained minimization without derivatives*, Mathematical Programming, 97 (2003), pp. 605–623.

References iv

L. Roberts and C. W. Royer, *Direct search based on probabilistic descent in reduced spaces*, SIAM Journal on Optimization, 33 (2023), pp. 3057–3082.

S. Shan, W. Ding, J. Passananti, H. Zheng, and B. Y. Zhao, *Prompt-specific poisoning attacks on text-to-image generative models*, arXiv preprint arXiv:2310.13828, (2023).

D. A. Spielman and S.-H. Teng, *Smoothed analysis of algorithms: Why the simplex algorithm usually takes polynomial time*, Journal of the ACM, 51 (2004), pp. 385–463.

S. F. B. Tett, J. M. Gregory, N. Freychet, C. Cartis, M. J. Mineter, and L. Roberts, *Does model calibration reduce uncertainty in climate projections?*, Journal of Climate, 35 (2022), pp. 2585–2602.

L. N. Vicente, *Worst case complexity of direct search*, EURO Journal on Computational Optimization, 1 (2013), pp. 143–153.