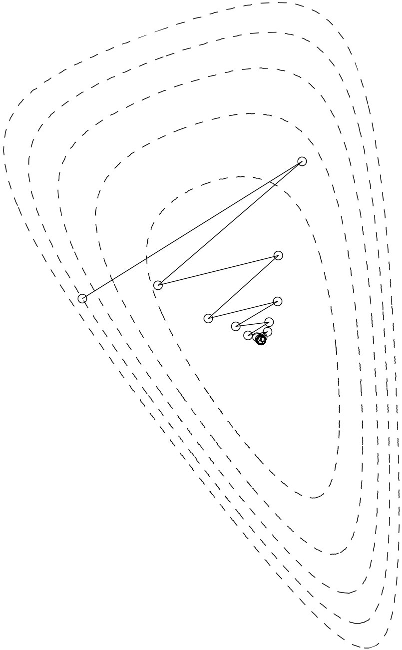

Optimization Algorithms

Derivative-Free (Black-Box) Optimization

Marc Toussaint
Technical University of Berlin
Winter 2024/25

Derivative-Free (Black-Box) Optimization

- Let $x \in \mathbb{R}^n$, $f: \mathbb{R}^n \to \mathbb{R}$, find

$$\underset{x}{\operatorname{argmin}} \; f(x)$$

- Derivative-Free/Blackbox optimization:
  - No access to ∇f or $\nabla^2 f$, sometimes also noisy evaluations $f(x)$

Derivative-Free (Black-Box) Optimization

- Let $x \in \mathbb{R}^n$, $f : \mathbb{R}^n \to \mathbb{R}$, find

$$\operatorname*{argmin}_x \; f(x)$$

- Derivative-Free/Blackbox optimization:
  - No access to ∇f or $\nabla^2 f$, sometimes also noisy evaluations $f(x)$
- Algorithms needs to collect *data* D about f, and decide on next queries

- Many variants:
  - Classical derivative-free, implicit filtering, model-based optimization
  - Heuristics: Nelder-Mead, Coordinate search, Twiddle, Pattern Search
  - Stochastic Search, evolution strategies, EDAs, other EAs
  - Bayesian Optimization, Global Optimization
  - others?

Implicit Filtering

- Estimates the local gradient using finite differencing

$$\nabla_\epsilon f(x) \approx \Big[\frac{1}{2\epsilon}(f(x+\epsilon e_i) - f(x-\epsilon e_i))\Big]_{i=1,..,n}$$

- Lines search along the gradient; if not succesful, decrease ϵ
- Can be extended by using $\nabla_\epsilon f(x)$ to update an approximation of the Hessian (as in BFGS)

Model-based optimization

following Nodecal et al. “Derivative-free optimization”

Model-based optimization

- The previous stochastic serach methods are heuristics to update θ

  Why not store the previous data directly?

- Model-based optimization takes the approach
  - Store a data set $\theta = D = \{(x_i, y_i)\}_{i=1}^n$ of previously explored points (let $\hat{x}$ be the current minimum in D)
  - Compute a (quadratic) model $D \mapsto \hat{f}(x) = \phi_2(x)^\top \beta$
  - Choose the next point as

$$x^+ = \operatorname*{argmin}_x \hat{f}(x) \;\text{ s.t. }\; |x - \hat{x}| < \alpha$$

  - Update D and α depending on $f(x^+)$

- The argmin is solved with constrained optimization methods

Model-based optimization

Initialize D with at least $\frac{1}{2}(n+1)(n+2)$ data points
repeat
  Compute a regression $\hat f(x) = \phi_2(x)^\top \beta$ on D
  Compute $x^+ = \mathrm{argmin}_x\, \hat f(x)$ s.t. $|x - \hat x| < \alpha$
  Compute the improvement ratio $\varrho = \frac{f(\hat x) - f(x^+)}{\hat f(\hat x) - \hat f(x^+)}$
  if $\varrho > \epsilon$ **then**
    Increase the stepsize α
    Accept $\hat x \leftarrow x^+$
    Add to data, $D \leftarrow D \cup \{(x^+, f(x^+))\}$
  else
    if $\det(D)$ is too small **then** *// Data improvement*
      Compute $x^+ = \mathrm{argmax}_x \det(D \cup \{x\})$ s.t. $|x - \hat x| < \alpha$
      Add to data, $D \leftarrow D \cup \{(x^+, f(x^+))\}$
    else
      Decrease the stepsize α
    end if
  end if
  Prune the data, e.g., remove $\mathrm{argmax}_{x \in \Delta} \det(D \setminus \{x\})$
until x converges

Model-based optimization

- Optimal parameters (with data matrix $X \in \mathbb{R}^{n \times \dim(\beta)}$)

$$\hat\beta^{\text{ls}} = (\boldsymbol{X}^\top \boldsymbol{X})^{\text{-1}} \boldsymbol{X}^\top y$$

  The determinant $\det(\boldsymbol{X}^\top \boldsymbol{X})$ or $\det(\boldsymbol{X})$ (denoted $\det(D)$ on the previous slide) is a measure for well the data supports the regression. The data improvement explicitly selects a next evaluation point to increase $\det(D)$.

- Nocedal describes in more detail a geometry-improving procedure to update D.

- Model-based optimization is closely related to Bayesian approaches. But
  - Should we really prune data to have only a minimal set D (of size $\dim(\beta)$?)
  - Is there another way to think about the "data improvement" selection of x^+? ($\to$ maximizing uncertainty/information gain)

Nelder-Mead method – Downhill Simplex Method

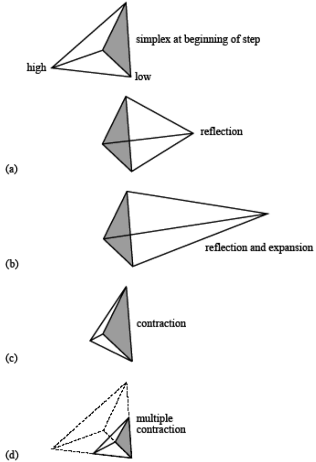

Figure 10.4.1. Possible outcomes for a step in the downhill simplex method. The simplex at the beginning of the step, here a tetrahedron, is shown, top. The simplex at the end of the step can be any one of (a) a reflection away from the high point, (b) a reflection and expansion away from the high point, (c) a contraction along one dimension from the high point, or (d) a contraction along all dimensions towards the low point. An appropriate sequence of such steps will always converge to a minimum of the function.

Nelder-Mead method – Downhill Simplex Method

- Let $x \in \mathbb{R}^n$
- Maintain $n+1$ points $x_0, .., x_n$, sorted by $f(x_0) < ... < f(x_n)$
- Compute center c of points
- Reflect: $y = c + \alpha(c - x_n)$
- If $f(y) < f(x_0)$: Expand: $y = c + \gamma(c - x_n)$
- If $f(y) > f(x_{n\text{-}1})$: Contract: $y = c + \varrho(c - x_n)$
- If still $f(y) > f(x_n)$: Shrink $\forall_{i=1,..,n} x_i \leftarrow x_0 + \sigma(x_i - x_0)$

- Typical parameters: $\alpha = 1, \gamma = 2, \varrho = -\frac{1}{2}, \sigma = \frac{1}{2}$

Coordinate Search

Input: Initial $x \in \mathbb{R}^n$

```
repeat
    for i = 1,..,n do
        α* = argmin_α f(x + α e_i)            // Line Search
        x ← x + α* e_i
    end for
until x converges
```

repeat
- **for** $i = 1, .., n$ **do**
  - $\alpha^* = \operatorname{argmin}_\alpha f(x + \alpha \boldsymbol{e}_i)$ *// Line Search*
  - $x \leftarrow x + \alpha^* \boldsymbol{e}_i$
- **end for**

until x converges

- The LineSearch must be approximated
  - E.g. abort on any improvement, when $f(x + \alpha \boldsymbol{e}_i) < f(x)$
  - Remember the last successful stepsize α_i for each coordinate

Twiddle

Input: Initial $x \in \mathbb{R}^n$, initial stepsizes α_i for all $i = 1 : n$

repeat
    for $i = 1, .., n$ **do**
        $x \leftarrow \text{argmin}_{y \in \{x - \alpha_i e_i, x, x + \alpha_i e_i\}} f(y)$ // *twiddle* x_i
        Increase α_i if x changed; decrease α_i otherwise
    end for
until x converges

Pattern Search

- In each iteration k, have a (new) set of search directions $D_k = \{d_{ki}\}$ and test steps of length α_k in these directions
- In each iteration, adapt the search directions D_k and step length α_k

Details: See Nocedal et al.