Automatic modeling along search directions for high dimensional optimization

Abstract Optimization and machine learning must tune models to fit data. Large-scale problems are typically optimized using a variant of gradient descent, where the gradient is calculated automatically, but the lack of higher-order information about function behavior slows progress. This poster explores the possibility of using unconventional number types to collect more "intelligence" about function behavior along a chosen search direction. By extracting a richer model of its behavior, it becomes easier to build robust and effective optimization routines.

Timothy E. Holy

Washington University in St. Louis

July 16, 2020

Automatic modeling, an extension of automatic differentiation

Traditional approach: compute gradients (and possibly Hessians) by automatic differentiation and build a model of the objective. Use model to predict next trial point. (But how far can you trust the model?)

Concept behind automatic differentiation

Dual numbers $a + b\epsilon$ where ϵ is an infinitesimal obeying $\epsilon^2 = 0$. Evaluate $f(x + w\epsilon) = f(x) + wf'(x)\epsilon$. **Julia packages**: ForwardDiff, Zygote (reverse mode).

Alternative approach: automatically model function with guaranteed limits.
Julia packages: IntervalArithmetic, TaylorSeries, *Bound3Numbers* (tentative name).

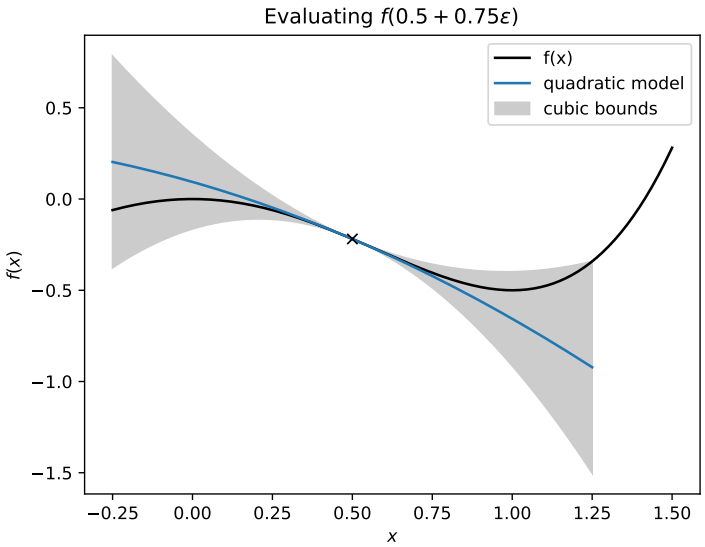
Concept behind automatic bounding

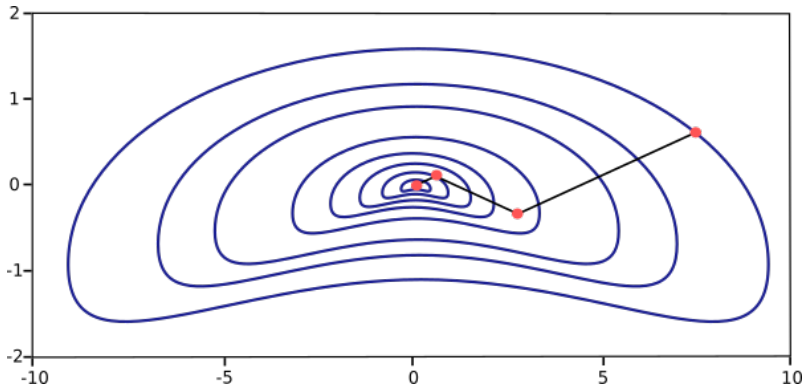
Model $f(y)$ for $y \in [x - w, x + w]$ as a second-order polynomial with third-order remainder, $|f(x + \alpha w) - f(x) - \alpha w f'(x) - \frac{1}{2}\alpha^2 w^2 f''(x)| \le r|\alpha|^3$, by evaluating $f(x + w\epsilon)$. Requires *Bound3 numbers* $v + s\epsilon + c\epsilon^2 + r\rho$, where

$$a\epsilon + b\epsilon = (a + b)\epsilon \qquad a\epsilon^2 + b\epsilon^2 = (a + b)\epsilon^2 \qquad a\rho + b\rho = (|a| + |b|)\rho$$

$$\epsilon^3 = \epsilon\rho = \rho^2 = \rho \quad (\text{models } |\alpha|^n \le |\alpha|^3 \text{ for } |\alpha| \le 1)$$

Using automatic bounding for optimization





Computing $f(\mathbf{x} + \omega\mathbf{p}\epsilon)$ enables simple & effective line searches:

Features:

- You have slope and curvature
- You have a guaranteed interval for improvement (no checking needed)
- $\Rightarrow$ Globally-convergent Newton's method in 1d with just polynomial computations.

$$\alpha = \begin{cases} \frac{-s}{c+\sqrt{c^2+3r|s|}} & c > 0; \\ -\operatorname{sign}(s)\frac{|c|+\sqrt{c^2+3r|s|}}{3r} & \end{cases}$$

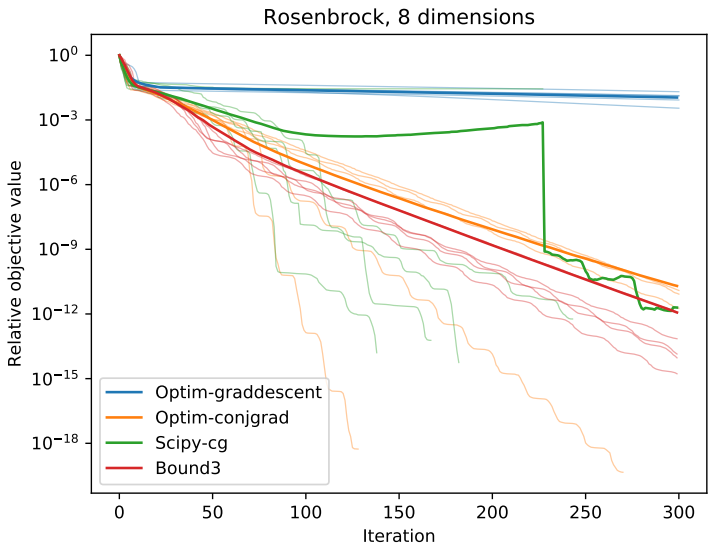
$$\mathbf{x} \to \mathbf{x} + \omega \operatorname{clamp}(\alpha, -1, 1)\mathbf{p}$$

$$\alpha_t = \frac{c^2\delta + 3r|s|/2}{c^2 + 3r|s|} \qquad (\delta \approx 10^{-5})$$

$$\omega \to \alpha/\alpha_t$$

Results & conclusions

The (multidimensional) Rosenbrock function can be defined as $f(\mathbf{x}) = \sum_i 100(x_{i+1} - x_i^2)^2 + (1 - x_i)^2$
Results from gradient descent and three different conjugate-direction methods:



200 different starting points `rand(8)`, average across all runs, a few individual traces shown in faint colors. Note that `scipy.fmin_cg` hops to wrong basin $\sim 0.5\%$ of the time.

Optimization with ϵ, ϵ^2, ρ numbers shares many features with more conventional methods. We trade increased complexity in the "automodeler" against decreased complexity in the algorithm.
Future directions:

- Stochastic descent: using the bounds to detect the impact of batches on updates
- Hybrid first- and second-order methods: extracting gradient and "scale"

Acknowledgements: David Sanders and Jarrett Revels for inspiration, and NIBIB UF1 NS108176 for funding.