Propagating Derivatives through Implicit Functions in Reverse Mode Autodiff

Johann Gaebler[1] Charles Margossian[2]

[1]Stanford University [2]Columbia University

Objectives

`Stan` offers reverse-mode autodiff for implicitly defined functions. We:

- Derive a novel adjoint method for propagating derivatives through implicit functions;
- Develop realistic use cases for implicit functions in statistics drawn from pharmacometrics;
- Show that the adjoint method offers speedup over currently implemented naïve method in all regimes.

Introduction

`Stan` is an open-source programming language designed for Bayesian data analysis [1]. `Stan`'s No-U-Turn Sampler, which efficiently samples the posterior distribution of even comparatively high-dimensional distributions [2], relies on the `Stan Math C++` library [3], which provides forward-mode and reverse-mode automatic differentiation ("autodiff") functionality.

The `Stan Math` library's functionality extends beyond simple use cases (e.g., addition and the exponential function) to complex, real-world applications, such as the solution of ordinary differential equations. This project implements efficient reverse-mode autodiff methods for the propagation of derivatives through implicit functions.

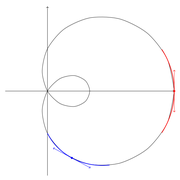

Figure: A limaçon trisectrix, defined implicitly by $x^2 + y^2 = (x^2 + y^2 - 2x)^2$. The variable y is a continuously differentiable function of x locally near the blue line, but not the red.

Setting

We define our implicit function as follows. Let $\mathbf{x} \in \mathbb{R}^n$ and $\mathbf{y} \in \mathbb{R}^m$ be related by

$$f(\mathbf{x}, \mathbf{y}) = \mathbf{0}. \tag{1}$$

where $f : \mathbb{R}^{n+m} \to \mathbb{R}^m$ is continuously differentiable. The implicit function theorem states that if the Jacobian $\frac{\partial f}{\partial \mathbf{y}}$ is invertible at a solution $(\mathbf{x}_0, \mathbf{y}_0)$, then $\mathbf{y}$ can be locally expressed as a continuously differentiable function of $\mathbf{x}$, and

$$\frac{\partial \mathbf{y}}{\partial \mathbf{x}} = -\left[\frac{\partial f}{\partial \mathbf{y}}\right]^{-1} \frac{\partial f}{\partial \mathbf{x}}. \tag{2}$$

Reverse-mode autodiff calculates the contraction of a Jacobian with a cotangent vector $\boldsymbol{\xi} \in \mathbb{R}^m$. Therefore, our algorithms will calculate the quantity

$$\boldsymbol{\xi}^\top \left[\frac{\partial \mathbf{y}}{\partial \mathbf{x}}\right]. \tag{3}$$

Algorithm 1

Algorithm 1: Naïve method.
Data: The implicit function $f(\mathbf{x}, \mathbf{y})$ and initial cotangent $\boldsymbol{\xi}$.
Result: The adjoint of the implicit function $\mathbf{x} \mapsto \mathbf{y}$ with respect to $\boldsymbol{\xi}$.

for $i = 1, \ldots, n$ **do**
  Calculate $\frac{\partial f}{\partial x_i}$. (One forward-mode pass.)
end
for $i = 1, \ldots, m$ **do**
  Calculate $\frac{\partial f}{\partial y_i}$. (One forward-mode pass.)
end
Calculate the LU-decomposition of $\frac{\partial f}{\partial \mathbf{y}}$.
for $i = 1, \ldots, n$ **do**
  Calculate $\mathbf{M}[:, i] = \left[\frac{\partial f}{\partial \mathbf{y}}\right]^{-1} \frac{\partial f}{\partial x_i}$. (One matrix solve.)
end
Calculate $\boldsymbol{\xi}_{\text{out}} = \boldsymbol{\xi}^\top \mathbf{M}$. (One matrix multiplication.)
return ξ_{out}.

Algorithm 2

Algorithm 2: Adjoint method.
Data: The implicit function $f(\mathbf{x}, \mathbf{y})$ and initial cotangent $\boldsymbol{\xi}$.
Result: The adjoint of the implicit function $\mathbf{x} \mapsto \mathbf{y}$ with respect to $\boldsymbol{\xi}$.

for $i = 1, \ldots, m$ **do**
  Calculate $\frac{\partial f}{\partial y_i}$. (One forward-mode pass.)
end
Calculate the LU-decomposition of $\frac{\partial f}{\partial \mathbf{y}}$.
Calculate $\boldsymbol{\eta}^\top = \boldsymbol{\xi}^\top \left[\frac{\partial f}{\partial \mathbf{y}}\right]^{-1}$. (One matrix solve.)
Calculate $\boldsymbol{\xi}_{\text{out}} = \boldsymbol{\eta}^\top \frac{\partial f}{\partial \mathbf{x}}$. (One nested reverse-mode pass.)
return ξ_{out}.

An example from Pharmacometrics

We consider a two-compartment model. Patient i consumes a dose d of a drug at intervals of length τ. The concentration of the drug in the main and peripheral chambers satisfies the ODE

$$y_1'(t) = -\kappa_{i,1} \cdot y_1(t), \tag{4}$$
$$y_2'(t) = \kappa_{i,1} \cdot y_1(t) - \kappa_{i,2} \cdot y_2(t). \tag{5}$$

The patient is at a "steady state" when

$$y_1(t) - y_1(t + \tau) = d, \tag{6}$$
$$y_2(t) - y_2(t + \tau) = 0. \tag{7}$$

Measurements are taken of concentration in the main chamber for each patient at steady state. Measurement $m_{i,k}$ taken at time t_k satisfies

$$m_{i,k} \sim \mathcal{N}(y_1(t_k), \sigma). \tag{8}$$

To simulate real-world use of implicit functions, we calculate the gradient of this likelihood function with respect to the parameters $\kappa_{1,1}$, $\kappa_{1,2}$, …, $\kappa_{n,1}$, $\kappa_{n,2}$ using both algorithms across a range of values of n. We test two conditions: *fixed parameter*, where $\kappa_{1,1} = \cdots = \kappa_{n,1}$ and $\kappa_{1,2} = \cdots = \kappa_{n,2}$; and *variable parameter* where each patient's parameters are free to vary independently of the other patients'.

Results

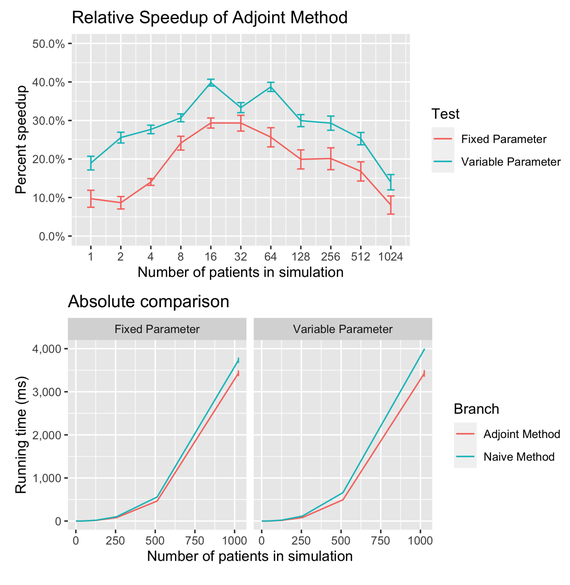

Figure: Speedup in the pharmacometric example using our adjoint method compared to the naïve method.

Conclusion

Our adjoint method offers speedup over the currently used naïve method of propagating derivatives through implicit functions. This speedup exists in all regimes, but grows larger as the number of parameters in the implicit function's defining relation grows larger. Asymptotically, however, the relative speedup decreases as the cost of LU decomposition dominates.

References

[1] Bob Carpenter, Andrew Gelman, Matthew D Hoffman, Daniel Lee, Ben Goodrich, Michael Betancourt, Marcus A Brubaker, Jiqiang Guo, Peter Li, and Allen Riddell. Stan: a probabilistic programming language. *Grantee Submission*, 76(1):1–32, 2017.

[2] Matthew D Hoffman and Andrew Gelman. The no-u-turn sampler: adaptively setting path lengths in hamiltonian monte carlo. *J. Mach. Learn. Res.*, 15(1):1593–1623, 2014.

[3] Bob Carpenter, Matthew D Hoffman, Marcus Brubaker, Daniel Lee, Peter Li, and Michael Betancourt. The stan math library: Reverse-mode automatic differentiation in c++. *arXiv preprint arXiv:1509.07164*, 2015.