Why "AD" in ADModel Builder?

ADMB

Automatic Differentiation Model Builder

ADMB Foundation

sibert@hawaii.edu

Outline

- Why are we interested in differentation?
- What is "automatic" about it?
- Automatic differentiation versus finite difference approximation
- What the AUTODIF library does

Differentiation finds maxima

Maximum Likelihood

Simple Example — Quadratic Regression

$$L(a,b) = f(x_1, x_2, x_3, \ldots, x_n | a,b) = \sum_{i=1}^{n} \left[y_i - (a + bx_i^2) \right]^2$$

$$\widehat{(a,b)} = \frac{\text{arg max}}{(a,b)} L(a,b)$$

$$\begin{aligned} \frac{\partial L}{\partial a} &= 2\sum_{i=1}^{n} \left(a + bx_i^2 - y_i \right) \\ \frac{\partial L}{\partial b} &= 2\sum_{i=1}^{n} x_i^2 \left(a + bx_i^2 - y_i \right) \end{aligned}$$

Automatic Differentiation

$$L_i(a,b) = \left[y_i - (a + bx_i^2)\right]^2$$

```
L(i) = pow(y(i)-(a+b*pow(x(i),2)),2);
```

1	$t_1 = x_i^2$	x_i^2
2	$t_2 = bt_1$	bx_i^2
3	$t_3 = a + t_2$	$a + bx_i^2$
4	$t_4 = y_i - t_3$	$y_i - (a + bx_i^2)$
5	$t_5 = t_4^2$	L_i

Derivative Chains

$$\frac{dL}{da} = \frac{dL}{dt_5} \cdot \frac{dt_5}{dt_4} \cdot \frac{dt_4}{dt_3} \cdot \frac{dt_3}{da} = 2(a + bx^2 - y)$$

$$\frac{dL}{db} = \frac{dL}{dt_5} \cdot \frac{dt_5}{dt_4} \cdot \frac{dt_4}{dt_3} \cdot \frac{dt_3}{dt_2} \cdot \frac{dt_2}{db} = 2x^2(a + bx^2 - y)$$

AUTODIF Algorithm — Reverse Mode AD

$$L_i(a,b) = \left[y_i - (a + bx_i^2)\right]^2$$

```
L(i) = pow(y(i)-pow(a+b*x(i),2),2);
```

1	$t_1 = x_i^2$	x_i^2
2	$t_2 = bt_1$	bx_i^2
3	$t_3 = a + t_2$	$a + bx_i^2$
4	$t_4 = y_i - t_3$	$y_i - (a + bx_i^2)$
5	$t_5 = t_4^2$	L_i

Derivative computation, $\tau_k = \frac{dt_{k+1}}{dt_k}$

	$\tau_5 = 1$	$\frac{\partial L}{\partial L}$
5	$\tau_4 = 2t_4\tau_5$	$2[y_i - (a + bx_i^2)]$
4	$\tau_3 = -\tau_4$	$2(a + bx_i^2 - y_i)$
	$\dot{y}_i = t_4$	
3	$\tau_2 = \tau_3$	$2(a + bx_i^2 - y_i)$
	$\dot{a} = \tau_3$	$2(a + bx_i^2 - y_i)$
2	$\tau_1 = b\tau_2$	
	$\dot{b} = t_1\tau_2$	$2x_i^2(a + bx_i^2 - y_i)$
1	$\dot{x}_i = 2x_i\tau_1$	

... but be careful!

$$k = \begin{cases} k_1 & \Delta Q < Q_T \\ k_2 & \Delta Q \geq Q_T \end{cases}$$

Where Q_T, k_1, and k_2 are model parameters, and $\Delta Q = f(k, \ldots)$ is state variable predicted by the model. Straightforward implementation of this assumption as

```
if (Q < QT)
  k = k1;
else
  k = k2;
```

breaks the derivative chain.

What to do about it?

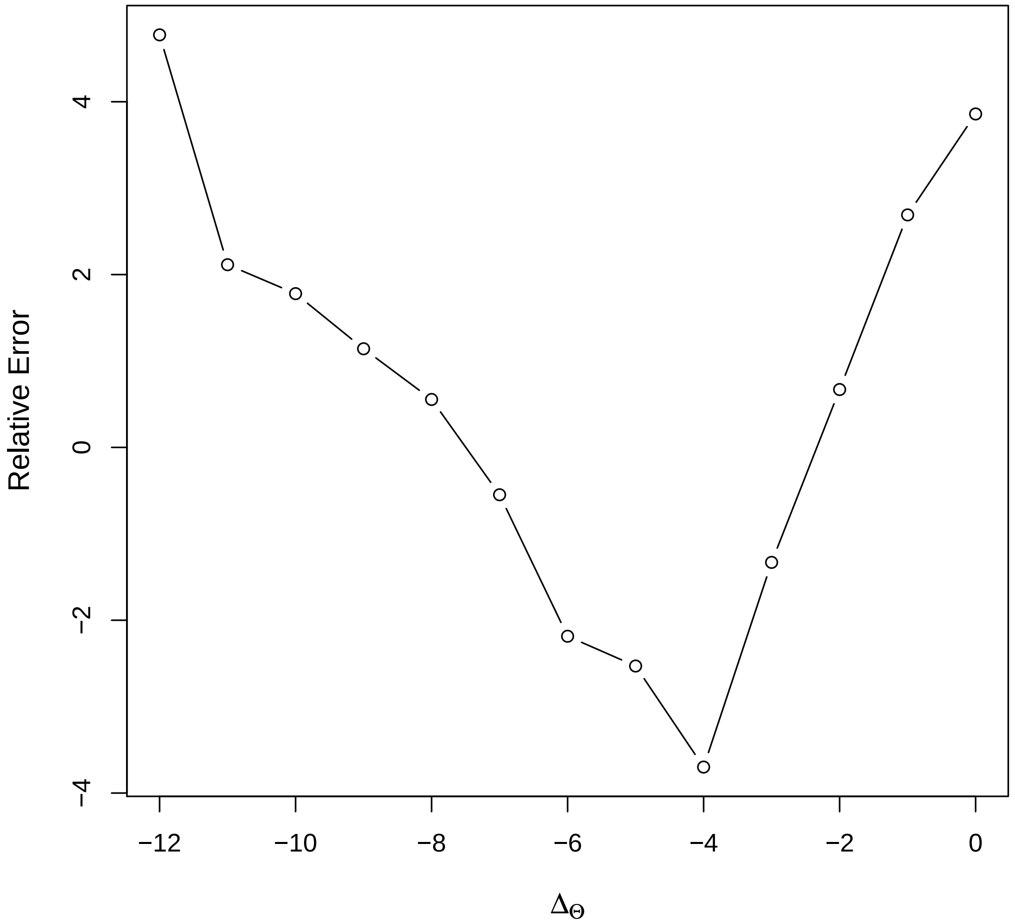
Finite difference approximations

- Expensive; cost proportional to number of parameters:

$$
\begin{aligned}
L = & \quad f(x_1, x_2, x_3, \ldots, x_n | \theta_1, \theta_2, \theta_3, \ldots, \theta_p) = & f(X|\Theta) \\
\frac{\partial L}{\partial \theta_j} \approx & \quad \frac{f(X|\theta_j) - f(X|\theta_j - \Delta_\theta)}{\Delta_\theta} & p+1 \text{ function evaluations} \\
\approx & \quad \frac{f(X|\theta_j + \Delta_\theta) - f(X|\theta_j - \Delta_\theta)}{2\Delta_\theta} & 2p \text{ function evaluations}
\end{aligned}
$$

- Inaccurate, at best an approximation.
- Requires computation of differences between numbers of the same order of magnitude; accumulates large round-off errors.

Finite Difference Errors



AUTODIF Library

- Analytically correct derivatives computed to same precision as objective function using the Chain Rule and "reverse mode" automatic differentiation
- C++ Library
- Classes for differentiable objects: scalars, vectors, matrices, higher dimensional arrays with flexible dimensions and optional subscript checking
- **All** operators $(+, -, \times, \div, \ldots)$ **and** mathematical functions (`sqrt()`, `exp()`, `log()`, `sin()`, `...`) overloaded
- Built-in derivative checker
- Efficient, stable quasi-Newton function minimizer; flexible convergence criteria
- Vector and matrix operations
- Built-in derivative checker

Exercise: the derivative checker (1)

- Compares AUTODIF chain rule derivatives with central finite-difference approximation.
- Invoke by:
  - Typing `-dd n` on the command line to start derivative checker after function evaluation `n`, or
  - by pressing `Ctrl C` during execution
- Specify which variable(s) you want checked.
- Specify the finite difference step size, 10^{-4} is a good place to start.

Exercise: the derivative checker (2)

- Introduce a derivative error in your code, for instance, by making the value of the regression coefficient depend on the dependant variable in using an `if` statement.
- Does the model converge?
- Check the derivatives using `-dd 1`.
- Fix the derivative error ...

References

Bard, Y. 1974. Nonlinear Parameter Estimation. Academic Press, San Diego.

Griewank, A. and G. F. Corliss (eds). 1992. Automatic differentiation of algorithms: theory, implementation, and application. Society of Industrial and Applied Mathematics.

Automatic Differentiation (Wikipedia)

Giles, M. B., D. Ghate, and M.C. Duta 2005. Using Automatic Differentiation for Adjoint CFD Code Development. Post SAROD Workshop-2005.
http://www2.maths.ox.ac.uk/~gilesm/psfiles/bangalore05.pdf