Why “AD” in ADModel Builder?
Outline

• Why are we interested in differentiation?
• 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

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

\[ (\hat{a}, \hat{b}) = \arg \max_{(a, b)} L(a, b) \]

\[ \frac{\partial L}{\partial a} = 2 \sum_{i=1}^{n} \left( a + bx_i^2 - y_i \right) \]

\[ \frac{\partial L}{\partial b} = 2 \sum_{i=1}^{n} x_i^2 \left( a + bx_i^2 - y_i \right) \]
Automatic Differentiation

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

\[ L(i) = \text{pow}(y(i)-(a+b*\text{pow}(x(i),2)),2); \]

1. \[ t_1 = x_i^2 \quad x_i^2 \]
2. \[ t_2 = bt_1 \quad bx_i^2 \]
3. \[ t_3 = a + t_2 \quad a + bx_i^2 \]
4. \[ t_4 = y_i - t_3 \quad y_i - (a + bx_i^2) \]
5. \[ t_5 = t_4^2 \quad L_i \]

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

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

\[
\begin{align*}
L(i) &= \text{pow}(y(i)-\text{pow}(a+b*x(i),2),2); \\
1 & \quad t_1 = x_i^2 \\
2 & \quad t_2 = bt_1 \\
3 & \quad t_3 = a + t_2 \\
4 & \quad t_4 = y_i - t_3 \\
5 & \quad t_5 = t_4^2
\end{align*}
\]

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

\[
\begin{align*}
\tau_5 &= 1 & \frac{\partial L}{\partial L} \\
5 & \quad \tau_4 = 2t_4 \tau_5 & 2[y_i - (a + bx_i^2)] \\
4 & \quad \tau_3 = -\tau_4 & 2(a + bx_i^2 - y_i) \\
3 & \quad \tau_2 = \tau_3 & 2(a + bx_i^2 - y_i) \\
2 & \quad \tau_1 = b \tau_2 & 2x_i^2(a + bx_i^2 - y_i) \\
1 & \quad \dot{x}_i = 2x_i \tau_1
\end{align*}
\]
... 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

```java
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:

\[ L = f(x_1, x_2, x_3, \ldots, x_n | \theta_1, \theta_2, \theta_3, \ldots, \theta_p) = f(X | \Theta) \]

\[
\frac{\partial L}{\partial \theta_j} \approx \frac{f(X | \theta_j) - f(X | \theta_j - \Delta \theta)}{\Delta \theta} \quad p + 1 \text{ function evaluations}
\]

\[
\approx \frac{f(X | \theta_j + \Delta \theta) - f(X | \theta_j - \Delta \theta)}{2 \Delta \theta} \quad 2p \text{ function evaluations}
\]

- 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

![Graph showing relative error vs. Δθ](image)
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 (+, −, ×, ÷, ...) 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


Automatic Differentiation (Wikipedia)

http://www2.maths.ox.ac.uk/~gilesm/psfiles/bangalore05.pdf