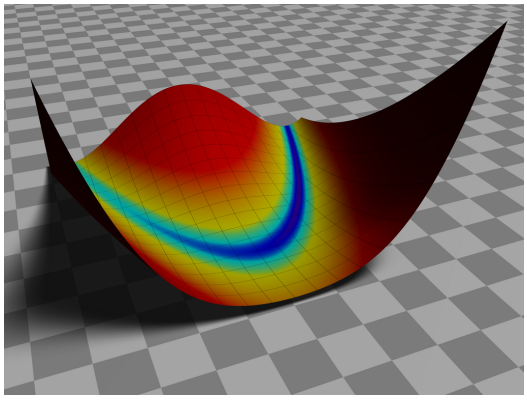
Automatic Differentiation

Lecture 10



ME EN 575
Andrew Ning
aning@byu.edu

Outline

Concept

- Symbolic Differentiation: Only useful with small functions. Usually impractical to represent a subroutine as a mathematical expression, and even if we did have symbolic expressions for all partial derivatives it would computationally ineffective as it can take an exponentially long time (lots of recomputing).
- Finite Difference: Very easy, but limited accuracy. Requires n function calls.
- Complex Step: A little more work, but analytic accuracy. Still requires n function calls (and each call is more expensive) so not effective for very large problems.

Automatic Differentiation: Analytic accuracy. Scales well. Extremely useful.

Overall Concept

Imagine a complex subroutine. If we break it down (which a computer does already) it is just a sequence of very simple elementary operations (sin, cos, exp, powers, etc.).

We know how to take the derivative of all of those elementary operations.

We also know how to apply the chain rule.

Thus, we can get numerically exact derivaties for really complex subroutines.

Mathematically, a program is just a bunch of elementary functions T_i, where each T_i is a function of all the variables that proceeded it

$$t_i = T_i(t_1, \ldots, t_{i-1})$$

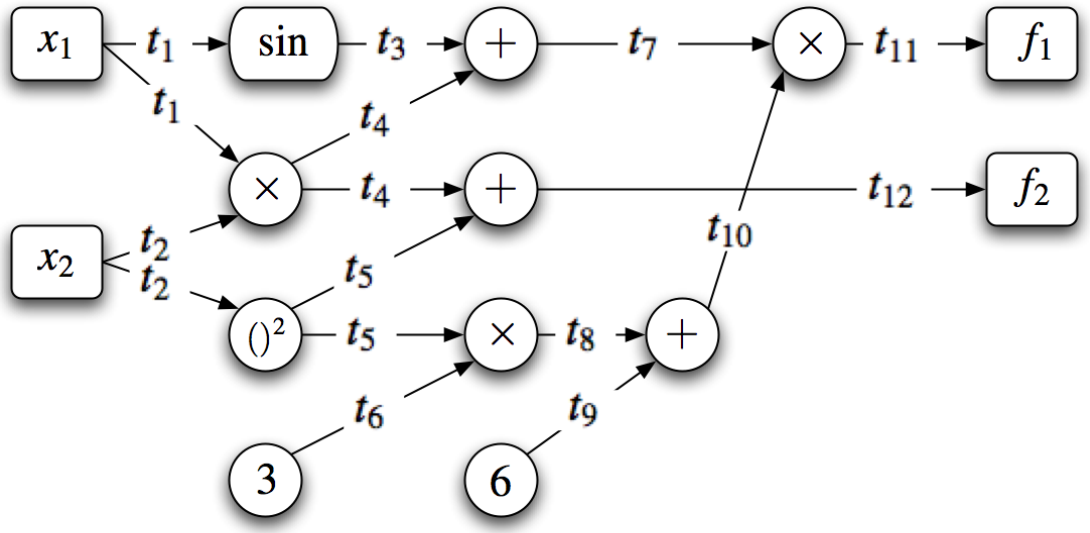
Apply chain rule:

$$\frac{\partial t_i}{\partial t_j} = \sum_{k=j}^{i-1} \frac{\partial T_i}{\partial t_k} \frac{\partial t_k}{\partial t_j} + \delta_{ij}, \quad \text{for} \quad j \leq i \leq n$$

Example:

$$f_1 = (x_1 x_2 + \sin x_1)(3x_2^2 + 6)$$
$$f_2 = x_1 x_2 + x_2^2$$

We can break this down into a sequence (on board).



Work out forward mode for

$$\frac{\partial f_1}{\partial x_1}$$

$$\frac{\partial t_{11}}{t_1} = t_{10}(\cos t_1 + t_2) = (3t_2^2 + g)(\cos t_1 + t_2) = \frac{f_1}{x_1}$$

Note that we can get derivative of other outputs with little additional cost:

$$\frac{\partial t_{12}}{t_1} = t_2 = \frac{f_2}{x_1}$$

Work out reverse mode for

$$\frac{\partial f_1}{\partial x_1}$$

Note that we get derivative w.r.t. other inputs for free:

$$\frac{\partial f_1}{\partial x_2}$$

If $n_{outputs} \gg n_{inputs} \Rightarrow$ use forward mode If $n_{inputs} \gg n_{outputs} \Rightarrow$ use reverse mode

Consider this simple example:

```
def func(x):
    y = x
    for i in range(100):
        y = sin(x + y)

    return y
```

Symbolic differentiation: really hard.

Automatic differentiation: easy.

Group Exercise: Get in groups of about 6 and line up in order.

Each of you is in charge of one computation. I'll get you started.

next computation	function value	derivative w.r.t t_1
$t_1 = x$	$t_1 = 0.1$	$\frac{dt_1}{dt_1} = 1.0$
$t_2 = t_1 + t_1$	$t_2 = 0.2$	$\frac{dt_2}{dt_1} = 2.0$

Example: symbolic vs AD

How can we use this more generally?

Two methods:

- Operator overloading
- Source code transformation

Source code transformation is generally faster, but more maintenance.

See Python notebook example.

Operator Overloading

Similar to complex step.

Consider a dual number $x + \epsilon dx$ with $\epsilon^2 = 0$

$$
\begin{aligned}
f(x) &= x_1 x_2 \\
f(x + \epsilon dx) &= (x_1 + \epsilon dx_1)(x_2 + \epsilon dx_2) \\
f(x + \epsilon dx) &= x_1 x_2 + \epsilon(x_1 dx_2 + x_2 dx_1) + \epsilon^2 (dx_1 dx_2) \\
f(x + \epsilon dx) &= x_1 x_2 + \epsilon(x_1 dx_2 + x_2 dx_1) \\
f(x + \epsilon dx) &= f(x) + \epsilon \frac{df}{dx}
\end{aligned}
$$

Complex step:

$$
\begin{aligned}
f(x) &= x_1 x_2 \\
f(x + ih) &= (x_1 + ih_1)(x_2 + ih_2) \\
f(x + ih) &= x_1 x_2 + i(x_1 h_2 + x_2 h_1) + i^2 (h_1 h_2) \\
f(x + ih) &= (x_1 x_2 - h_1 h_2) + i(x_1 h_2 + x_2 h_1)
\end{aligned}
$$

set $h_2 = 0$ and set $h_1 = 1 \times 10^{-20}$ (small h_1 necessary for higher-order terms that may exist)

$$
\frac{\partial f}{\partial x_1} = \frac{\mathrm{Im}[f(x + ih_1)]}{h_1}
$$

repeat for $h_1 = 0$