Automatic Differentiation

Benoît Legat

☐ Full Width Mode ☐ Present Mode

Table of Contents

Differentiation approaches

Chain rule

Forward Differentiation

Reverse differentiation

Comparison

Discontinuity

Differentiation approaches

We can compute partial derivatives in different ways:

1. **Symbolically**, by fixing one of the variables and differentiating with respect to the others, either manually or using a computer.
2. **Numerically**, using the formula $f'(x) \approx (f(x+h) - f(x))/h$.
3. **Algorithmically**, either forward or reverse : this is what we will explore here.

Chain rule

Consider $f(x) = f_3(f_2(f_1(x)))$. If we don't have the expression of f_1 but we can only evaluate $f_i(x)$ or $f'(x)$ for a given x ? The chain rule gives

$$f'(x) = f'_3(f_2(f_1(x))) \cdot f'_2(f_1(x)) \cdot f'_1(x).$$

Let's define $s_0 = x$ and $s_k = f_k(s_{k-1})$, we now have:

$$f'(x) = f'_3(s_2) \cdot f'_2(s_1) \cdot f'_1(s_0).$$

Two choices here:

$$\begin{array}{ll} \text{Forward} & \text{Reverse} \\ t_0 = 1 & r_3 = 1 \\ t_k = f'_k(s_{k-1}) \cdot t_{k-1} & r_k = r_{k+1} \cdot f'_{k+1}(s_k) \end{array}$$

Forward Differentiation

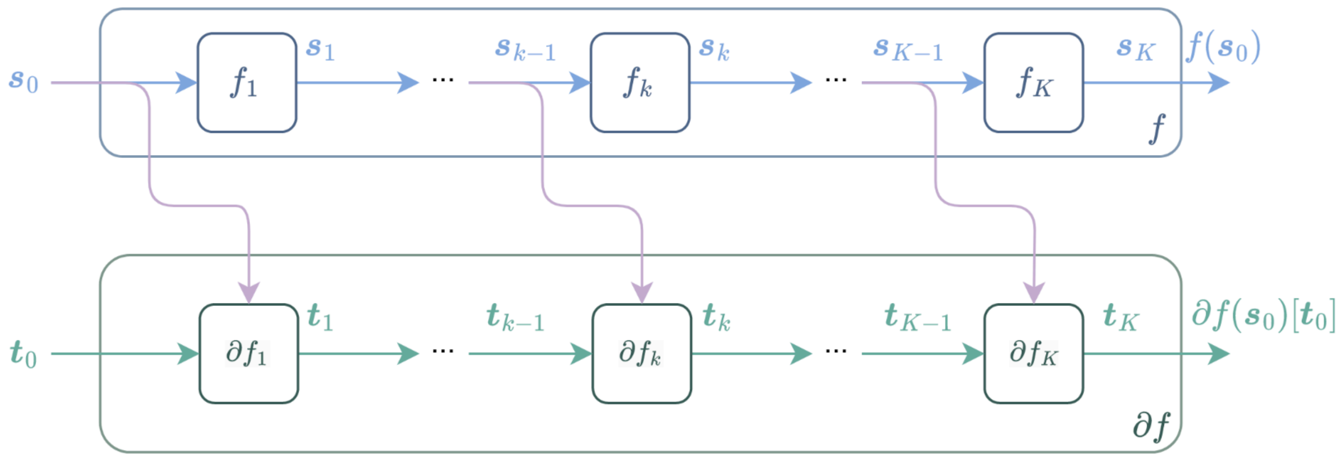


Figure 8.1

Implementation

```
struct Dual{T}
    value::T # s_k
    derivative::T # t_k
end
```

```
Base.:-(x::Dual{T}) where {T} = Dual(-x.value, -x.derivative)
```

```
Base.:*(x::Dual{T}, y::Dual{T}) where {T} = Dual(x.value * y.value, x.value *
y.derivative + x.derivative * y.value)
```

```
Dual(-3, -10)
```

```
-Dual(1, 2) * Dual(3, 4)
```

```
f_1 (generic function with 1 method)
```

```
f_1(x, y) = x * y
```

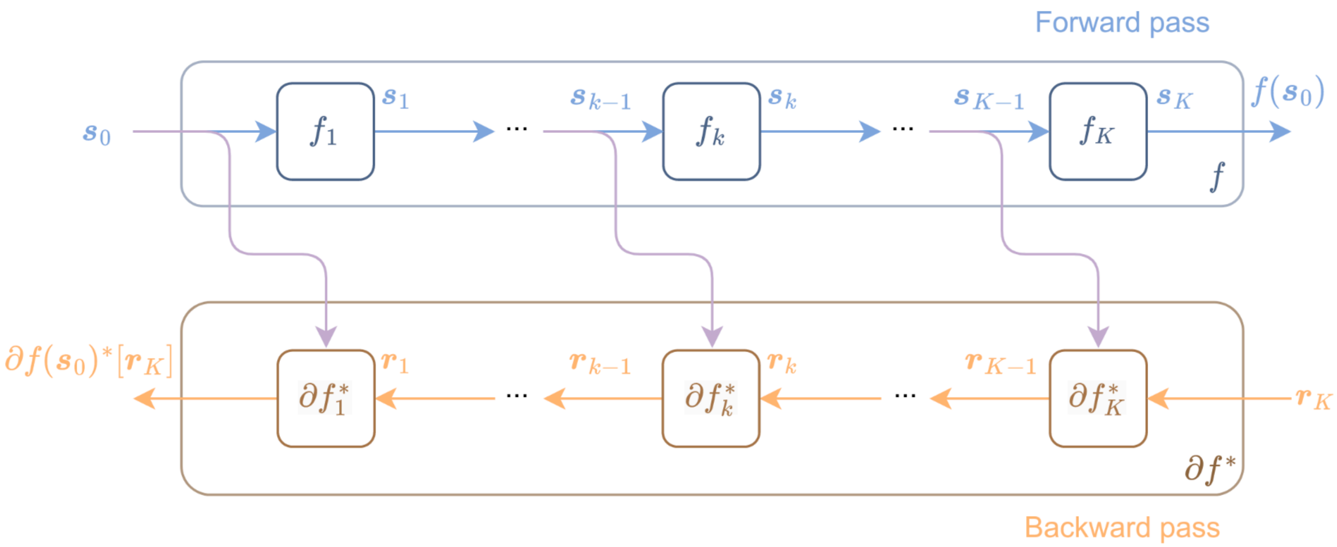
```
f_2 (generic function with 1 method)
```

```
f_2(s1) = -s1
```

Dual(-3, -10)

```
(f_2 ∘ f_1)(Dual(1, 2), Dual(3, 4))
```

Reverse differentiation



Two different takes on the multivariate chain rule

The chain rule gives us

$$\frac{\partial f_3}{\partial x}(f_1(x), f_2(x)) = \frac{\partial f_3}{\partial s_1}(s_1, s_2) \cdot \frac{\partial s_1}{\partial x} + \frac{\partial f_3}{\partial s_2}(s_1, s_2) \cdot \frac{\partial s_2}{\partial x}$$

To compute this expression, we need the values of $g(x)$ and $h(x)$ as well as the derivatives $\partial g/\partial x$ and $\partial h/\partial x$.

Forward

$$t_3 = \frac{\partial s_3}{\partial s_1} t_1 + \frac{\partial s_3}{\partial s_2} t_2$$

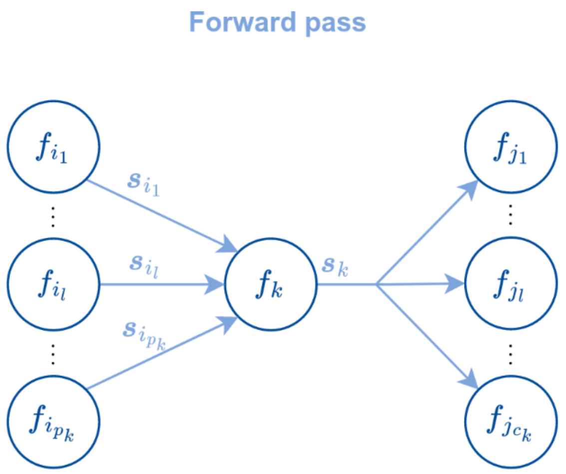
- Given s_1, s_2, computes $\frac{\partial s_3}{\partial s_1}(s_1, s_2)$ and $\frac{\partial s_3}{\partial s_2}(s_1, s_2)$
- Given t_1 and t_2, computes $\partial f_3/\partial x$

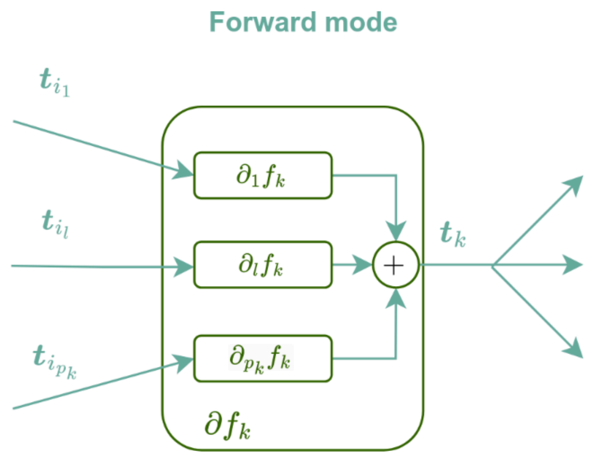
Reverse

- Given s_1, s_2, computes $\frac{\partial s_3}{\partial s_1}(s_1, s_2)$ and $\frac{\partial s_3}{\partial s_2}(s_1, s_2)$
- Given $r_3 = \partial s_K/\partial s_3$
  - **Add** $r_3 \cdot (\partial s_3/\partial s_1)$ to r_1
  - **Add** $r_3 \cdot (\partial s_3/\partial s_2)$ to r_2

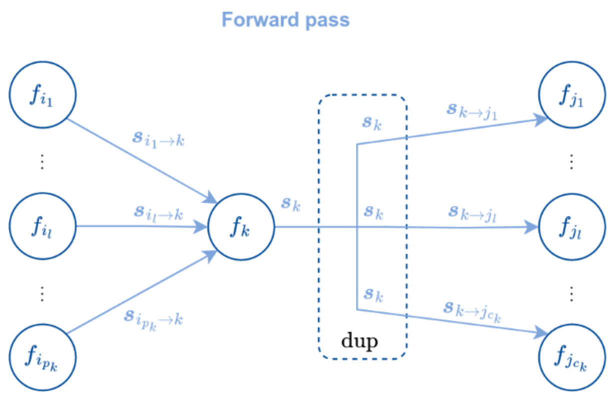
▶ **Apply this to $f_3(s_1, s_2) = s_1 + s_2$, $f_1(x) = x$ and $f_2(x) = x^2$**

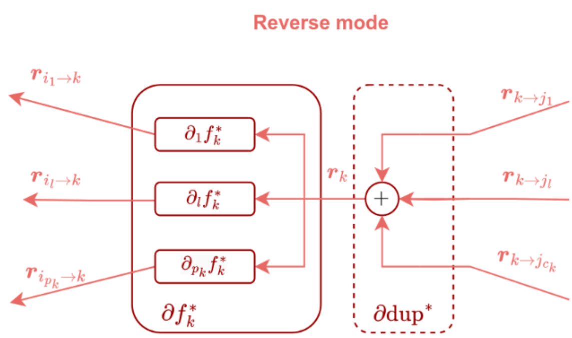
Forward tangents





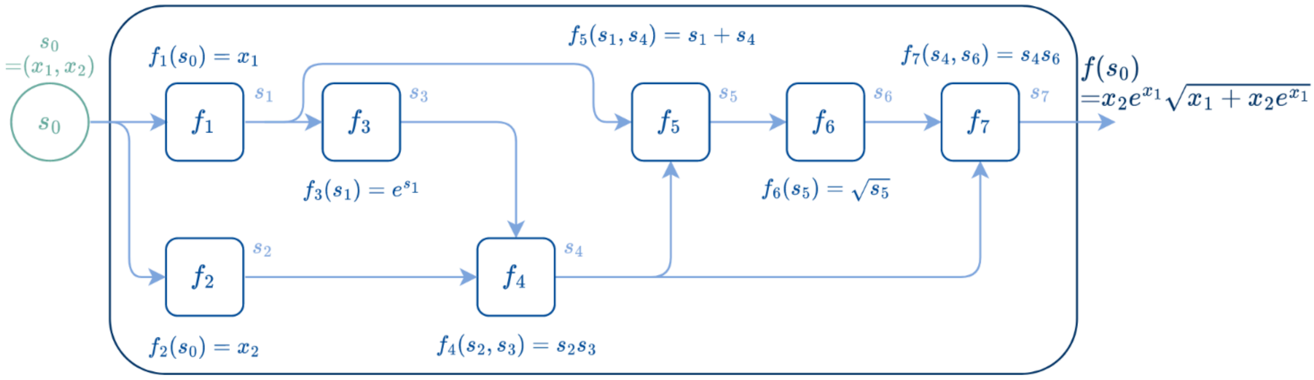
Reverse tangents





▶ **Why is $\partial \mathrm{dup}^*$ a sum ?**

Expression graph



▶ **Can this directed graph have cycles ?**

▶ **What happens if f_4 is handled before f_5 in the backward pass ?**

▶ **How to prevent this from happening ?**

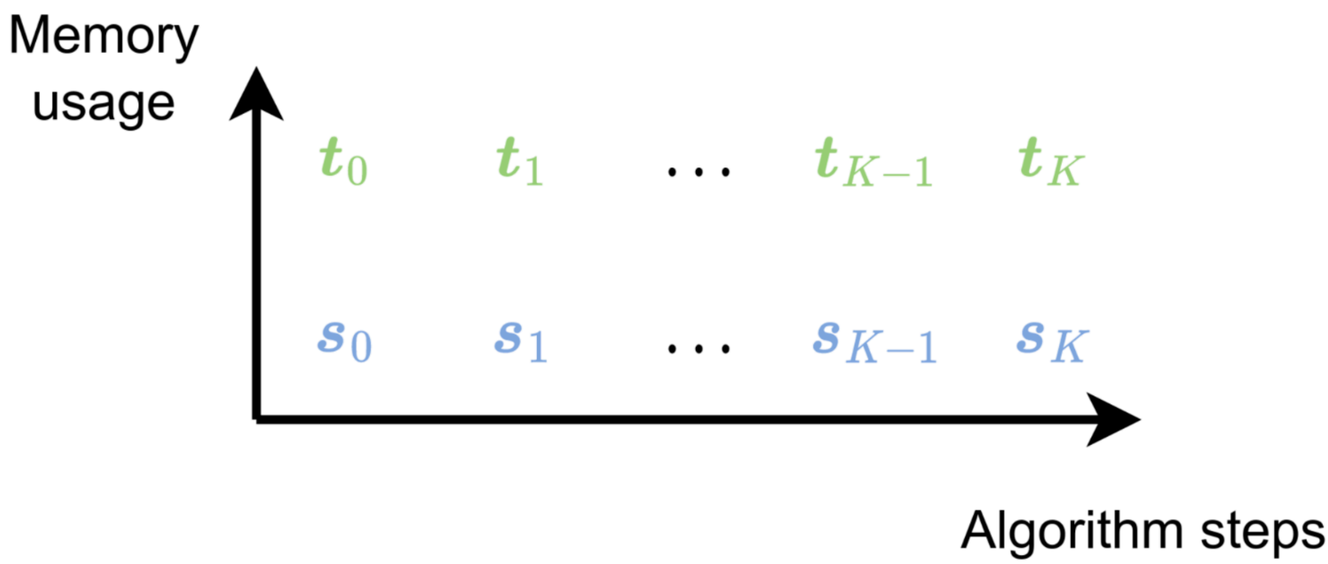
Comparison

- Forward mode of $f(x)$ with dual numbers `Dual.(x, v)` computes Jacobian-Vector Product (JVP) $J_f(x) \cdot v$
- Reverse mode of $f(x)$ computes Vector-Jacobian Product (VJP) $v^\top J_f(x)$ or in other words $J_v(x)^\top v$

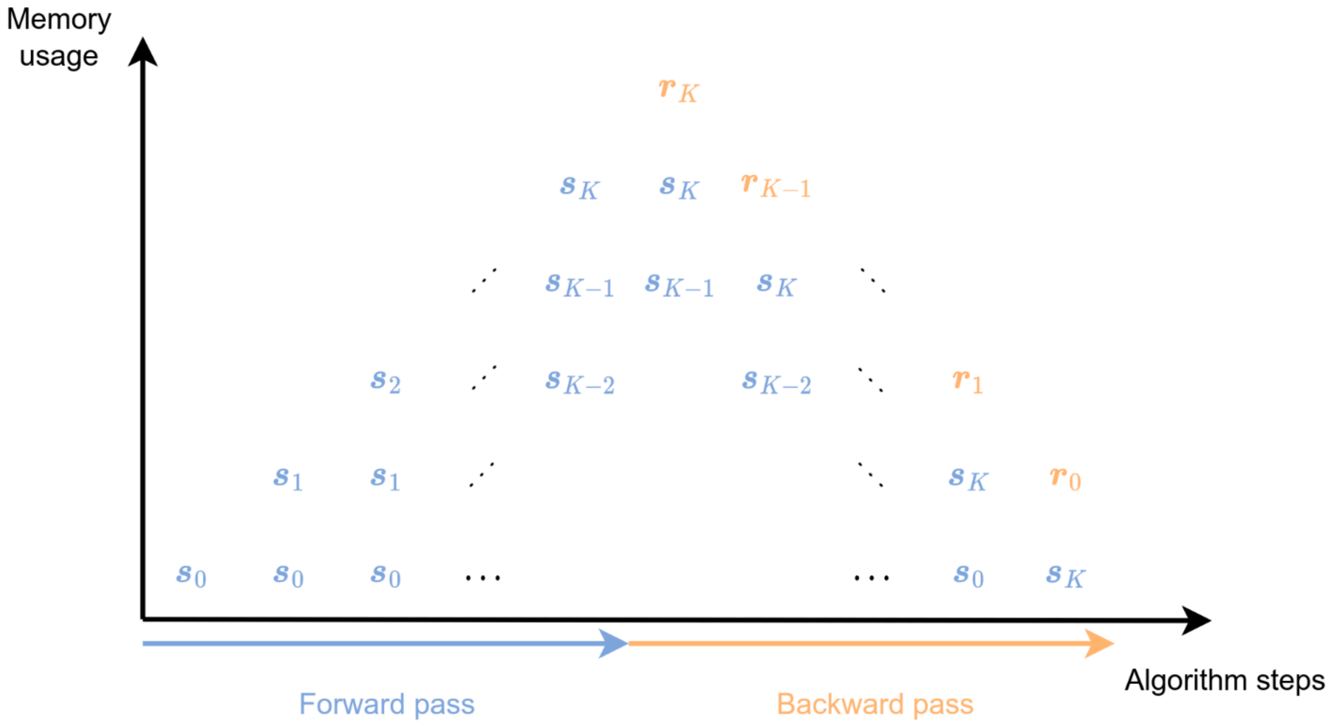
▶ **How can we compute the full Jacobian ?**

▶ **When is each mode faster than the other one to compute the full Jacobian ?**

Memory usage of forward mode



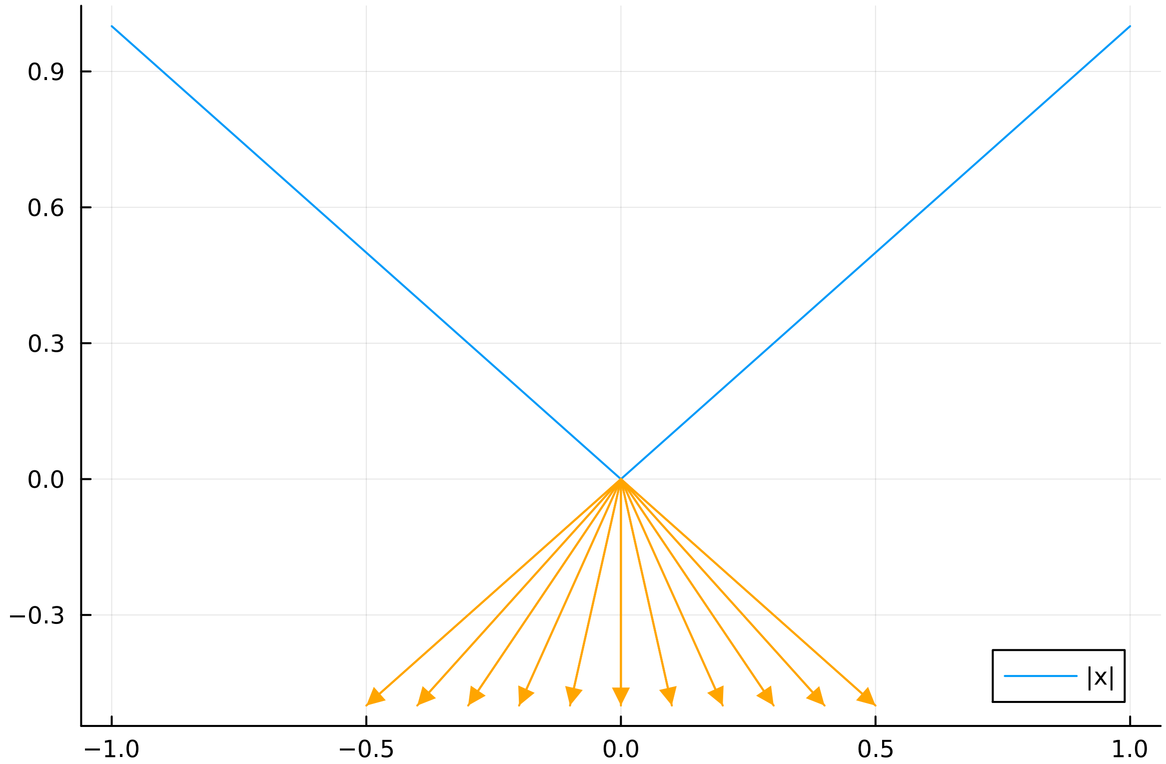
Memory usage of reverse mode



Discontinuity

The function $|x|$ is not differentiable at $x = 0$. If we approach from the left (that is, $x < 0$, the function is $-x$), then the derivative is -1. If we approach from the right (that is, $x > 0$, the function is x), then the derivative is 1. There is no valid gradient!

However, any number between -1 and 1 is a valid **subgradient**! Whereas the gradient is the normal to the **unique** tangent, the subgradient is an element of the **tangent cone**. Which one should we return ?



Forward mode

abs (generic function with 1 method)

```
abs(x) = ifelse(x < 0, -x, x)
```

abs_bis (generic function with 1 method)

```
abs_bis(x) = ifelse(x > 0, x, -x)
```

```julia
Base.isless(x::Dual, y::Real) = isless(x.value, y)
```

```julia
Base.isless(x::Real, y::Dual) = isless(x, y.value)
```

```
Dual(0, 1)
```

```julia
abs(Dual(0, 1))
```

```
Dual(0, -1)
```

```julia
abs_bis(Dual(0, 1))
```

Acknowledgements and further readings

- `Dual` is inspired from ForwardDiff
- `Node` is inspired from micrograd
- Here is a good intro to AD
- Figures are from the The Elements of Differentiable Programming book

The End

Utils

```
import MLJBase, Colors, Tables
```

```
using Graphs, GraphPlot, Printf
```

```
using Plots, PlutoUI, PlutoUI.ExperimentalLayout, HypertextLiteral; @htl, @htl_str
PlutoTeachingTools
```

```
img (generic function with 3 methods)
```

```
qa (generic function with 2 methods)
```