

# Automatic differentiation

Mauricio A. Álvarez

Machine Learning and Adaptive Intelligence  
The University of Sheffield



The  
University  
Of  
Sheffield.

# Contents

Derivatives and ways to compute them

AD modes

Forward mode

Reverse mode

Implementations

# Derivatives

- ❑ Derivatives are required to perform optimisation in several ML algorithms.
- ❑ For example, computing the gradient is necessary for **batch gradient descent** and **SGD**.
- ❑ Derivatives are also necessary for computing Hessians which are used in second-order optimisation methods.

# Methods to compute derivatives in computer programs

- ❑ **Manually** working out derivatives and coding them.
- ❑ Numeric differentiation using finite difference approximations.
- ❑ Symbolic differentiation.
- ❑ Automatic differentiation (or algorithmic differentiation).

# Example

- Suppose we have the following function

$$f(x, y) = x^2y + y + 2.$$

- We require to compute the gradient of this function, for example, because we want to use it in gradient descent,

$$\frac{df(x, y)}{d\mathbf{z}} = \begin{bmatrix} \frac{\partial f(x, y)}{\partial x} \\ \frac{\partial f(x, y)}{\partial y} \end{bmatrix},$$

where  $\mathbf{z} = [x \ y]^\top$ .

# Manual differentiation (I)

- We use our calculus knowledge to derive the proper equation.
  
- For the function we saw before, we need to apply the following rules of calculus
  - The derivative of a constant is 0.
  - The derivative of  $ax$  with respect to  $x$  is  $a$ , where  $a$  is a constant.
  - The derivative of  $x^a$  is  $ax^{a-1}$ .
  - The derivative is a linear operation so, the derivative of the sum of two functions is the sum of the derivatives.
  - The derivative of a constant times a function, is equal to the constant times the derivative of that function.
  
- Using these rules we get the following partial derivatives,

## Manual differentiation (II)

- Partial derivative of  $f(x, y)$  with respect to  $x$

$$\frac{\partial}{\partial x} f(x, y) = \frac{\partial}{\partial x} (x^2 y + y + 2) = 2xy.$$

- Partial derivative of  $f(x, y)$  with respect to  $y$

$$\frac{\partial}{\partial y} f(x, y) = \frac{\partial}{\partial y} (x^2 y + y + 2) = x^2 + 1.$$

- We can then write

$$\frac{df(x, y)}{d\mathbf{z}} = \begin{bmatrix} \frac{\partial f(x, y)}{\partial x} \\ \frac{\partial f(x, y)}{\partial y} \end{bmatrix} = \begin{bmatrix} 2xy \\ x^2 + 1 \end{bmatrix}$$

# Problems with manual differentiation

When a function  $f(\cdot)$  depends on many variables or is a rather complicated expression, manual differentiation is **tedious** and prone to mistakes.



# Finite difference approximations (I)

- Remember the definition of a derivative of a function  $h(x)$  at a point  $x_0$ ,

$$\begin{aligned}\frac{dh(x_0)}{dx} &= \lim_{x \rightarrow x_0} \frac{h(x) - h(x_0)}{x - x_0} \\ &= \lim_{\epsilon \rightarrow 0} \frac{h(x_0 + \epsilon) - h(x_0)}{\epsilon}\end{aligned}$$

- The partial derivative of  $h(x, y)$  at point  $(x_0, y_0)$  is defined as

$$\begin{aligned}\frac{\partial h(x_0, y_0)}{\partial x} &= \lim_{\epsilon \rightarrow 0} \frac{h(x_0 + \epsilon, y_0) - h(x_0, y_0)}{\epsilon} \\ \frac{\partial h(x_0, y_0)}{\partial y} &= \lim_{\epsilon \rightarrow 0} \frac{h(x_0, y_0 + \epsilon) - h(x_0, y_0)}{\epsilon}\end{aligned}$$

# Finite difference approximations (II)

```
def f(x, y):  
    return x**2*y + y + 2  
  
x_0 = 3  
y_0 = 2  
epsilon = 1e-6  
dfdx_numerical = (f(x_0+epsilon, y_0) - f(x_0, y_0))/epsilon  
dfdy_numerical = (f(x_0, y_0+epsilon) - f(x_0, y_0))/epsilon  
  
dfdx_analytical = 2*x_0*y_0  
dfdy_analytical = x_0**2 + 1
```

Script in python for the finite differences

```
In [22]: dfdx_numerical  
Out[22]: 12.000002001855137
```

```
In [23]: dfdx_analytical  
Out[23]: 12
```

$$\frac{\partial f(x,y)}{\partial x}$$

```
In [24]: dfdy_numerical  
Out[24]: 10.000000003174137
```

```
In [25]: dfdy_analytical  
Out[25]: 10
```

$$\frac{\partial f(x,y)}{\partial y}$$

# Problems with finite difference approximation

- ❑ The result is imprecise and gets worse with more complicated functions.
- ❑ We need to call the function at least twice. For big parametric models, we'd need to call the function several times becoming very inefficient.
- ❑ The method is easy to implement, so one can use it to test whether the **manual** implementation is correct.

# Symbolic differentiation (I)

- ❑ Symbolic differentiation performs an automatic manipulation of expressions to obtain the corresponding derivative expressions.
- ❑ The mathematical expression is represented using **data structures** (e.g. trees, lists, etc.).
- ❑ It is then possible to follow a mechanistic process to obtain the derivatives.

# Symbolic differentiation (II)



Mathematica



Maxima



Maple

# Symbolic differentiation with Mathematica

In[33]:= **D**[**x**, **x**]

Out[33]= 1

In[34]:= **D**[**4 x (1 - x)**, **x**]

Out[34]= 4 (1 - x) - 4 x

In[35]:= **D**[**16 x (1 - x) ((1 - 2 x) ^ 2)**, **x**]

Out[35]= 16 (1 - 2 x)<sup>2</sup> (1 - x) - 16 (1 - 2 x)<sup>2</sup> x - 64 (1 - 2 x) (1 - x) x

In[36]:= **D**[**64 x (1 - x) ((1 - 2 x) ^ 2) ((1 - 8 x + 8 x ^ 2) ^ 2)**, **x**]

Out[36]= 128 (1 - 2 x)<sup>2</sup> (1 - x) x (-8 + 16 x) (1 - 8 x + 8 x<sup>2</sup>) +  
64 (1 - 2 x)<sup>2</sup> (1 - x) (1 - 8 x + 8 x<sup>2</sup>)<sup>2</sup> - 64 (1 - 2 x)<sup>2</sup> x (1 - 8 x + 8 x<sup>2</sup>)<sup>2</sup> -  
256 (1 - 2 x) (1 - x) x (1 - 8 x + 8 x<sup>2</sup>)<sup>2</sup>

# Problems with symbolic differentiation

- Due to the mechanistic approach, there is usually a lot of **redundancy** in the expressions generated.
- If not handled properly, it produces unnecessary long expressions difficult to make sense of and to evaluate.
- Such behavior is known as *expression swell*.

## Example of *expression swell*

$n$	$l_n$	$\frac{dl_n}{dx}$
1	$x$	1
2	$4x(1-x)$	$4(1-x) - 4x$
3	$16x(1-x)(1-2x)^2$	$16(1-2x)^2(1-x) - 16(1-2x)^2x - 64(1-2x)(1-x)x$
4	$64x(1-x)(1-2x)^2(8x^2 - 8x + 1)^2$	$128(1-2x)^2(1-x)x(-8+16x)(1-8x+8x^2) + 64(1-2x)^2(1-x)(1-8x+8x^2)^2 - 64(1-2x)^2x(1-8x+8x^2)^2 - 256(1-2x)(1-x)x(1-8x+8x^2)^2$

**Expression swell**

Logistic map  $l_n = 4l_n(1 - l_n)$ ,  $l_1 = x$ .



# Simplify with Mathematica

In[40]:=  $D[16 x (1 - x) ((1 - 2 x)^2), x]$

Out[40]=  $16 (1 - 2 x)^2 (1 - x) - 16 (1 - 2 x)^2 x - 64 (1 - 2 x) (1 - x) x$

In[38]:=  $\text{Simplify}[16 (1 - 2 x)^2 (1 - x) - 16 (1 - 2 x)^2 x - 64 (1 - 2 x) (1 - x) x]$

Out[38]=  $-16 (-1 + 10 x - 24 x^2 + 16 x^3)$

In[41]:=  $D[64 x (1 - x) ((1 - 2 x)^2) ((1 - 8 x + 8 x^2)^2), x]$  **Expression swell**

Out[41]=  $128 (1 - 2 x)^2 (1 - x) x (-8 + 16 x) (1 - 8 x + 8 x^2) +$   
 $64 (1 - 2 x)^2 (1 - x) (1 - 8 x + 8 x^2)^2 - 64 (1 - 2 x)^2 x (1 - 8 x + 8 x^2)^2 -$   
 $256 (1 - 2 x) (1 - x) x (1 - 8 x + 8 x^2)^2$

In[42]:=  $\text{Simplify}[128 (1 - 2 x)^2 (1 - x) x (-8 + 16 x) (1 - 8 x + 8 x^2) +$   
 $64 (1 - 2 x)^2 (1 - x) (1 - 8 x + 8 x^2)^2 - 64 (1 - 2 x)^2 x (1 - 8 x + 8 x^2)^2 -$   
 $256 (1 - 2 x) (1 - x) x (1 - 8 x + 8 x^2)^2]$

Out[42]=  $-64 (-1 + 42 x - 504 x^2 + 2640 x^3 - 7040 x^4 + 9984 x^5 - 7168 x^6 + 2048 x^7)$

# Automatic differentiation

- AD is concerned about exact numerical computation of the derivatives, rather than their actual symbolic form.
- It computes the derivative by **only storing the values** of intermediate sub-expressions.
- It uses a combination of: **symbolic differentiation at the elementary operation level** and **keeping intermediate numerical results**.

# Contents

Derivatives and ways to compute them

AD modes

Forward mode

Reverse mode

Implementations

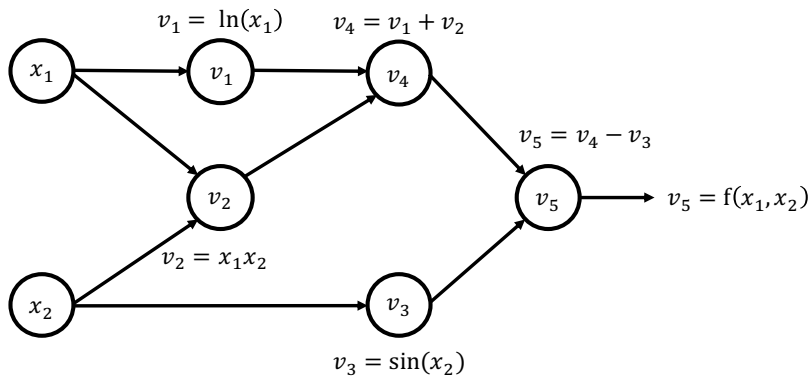
# Evaluation trace

- ❑ *Evaluation trace*: composition of elementary operations that lead to a full expression.
- ❑ As an example, let us consider the function

$$f(x_1, x_2) = \ln(x_1) + x_1 x_2 - \sin(x_2).$$

- ❑ The inputs are  $x_1$  and  $x_2$ .
- ❑ The elementary operations include
  - $v_1 = \ln(x_1)$
  - $v_2 = x_1 x_2$
  - $v_3 = \sin(x_2)$
  - $v_4 = v_1 + v_2$
  - $v_5 = v_4 - v_3$
  - $f(x_1, x_2) = v_5$ .

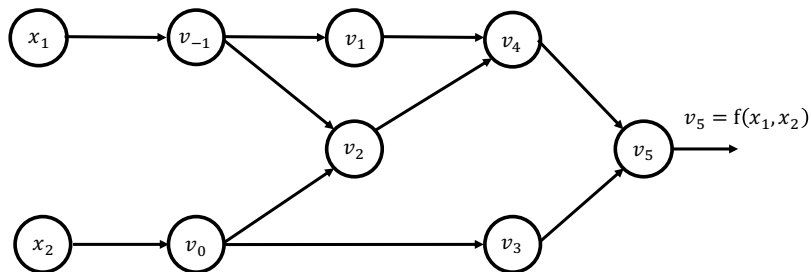
# Computational graph



# General notation

- Let  $\mathbf{f} : \mathbb{R}^n \rightarrow \mathbb{R}^m$ .
- Variables  $v_{i-n} = x_i$ , where  $i = 1, \dots, n$  are the input variables.
- Variables  $v_i$ , with  $i = 1, \dots, l$  are the intermediate variables.
- Variables  $y_{m-i} = v_{l-i}$ , with  $i = m - 1, \dots, 0$  are the output variables.

# New computational graph



# Jacobian

- Say that we have several functions  $y_i = f_i(\cdot)$  for  $i = 1, \dots, m$  that depend on several input variables  $x_1, x_2, \dots, x_n$ ,

$$\begin{aligned}y_1 &= f_1(x_1, \dots, x_n) \\y_2 &= f_2(x_1, \dots, x_n) \\&\vdots \\y_m &= f_m(x_1, \dots, x_n)\end{aligned}$$

- The Jacobian  $\mathbf{J}$  of dimensions  $m \times n$  is a matrix with entries  $\mathbf{J}_{ij} = \frac{\partial f_i}{\partial x_j}$  given as

$$\mathbf{J} = \begin{bmatrix} \frac{\partial f_1}{\partial x_1} & \frac{\partial f_1}{\partial x_2} & \cdots & \frac{\partial f_1}{\partial x_n} \\ \frac{\partial f_2}{\partial x_1} & \frac{\partial f_2}{\partial x_2} & \cdots & \frac{\partial f_2}{\partial x_n} \\ \vdots & \vdots & \vdots & \vdots \\ \frac{\partial f_m}{\partial x_1} & \frac{\partial f_m}{\partial x_2} & \cdots & \frac{\partial f_m}{\partial x_n} \end{bmatrix}$$



# Contents

Derivatives and ways to compute them

AD modes

Forward mode

Reverse mode

Implementations

# Forward accumulation mode

- Forward accumulation mode or tangent linear mode.
- To compute the derivative of  $f$  with respect to  $x_1$ , each intermediate variable  $v_i$  has a derivative

$$\dot{v}_i = \frac{\partial v_i}{\partial x_1}$$

- For each evaluation (or forward primal) trace, it builds a forward *derivative* (or tangent) trace.
- Essentially, this forward derivative trace is just implementing the *chain rule of differentiation*

$$\frac{dy}{dx} = \frac{dy}{du} \frac{du}{dz} \frac{dz}{dx}.$$

# Forward primal trace and tangent trace for $\frac{\partial y}{\partial x_1}$ (I)

- Let us compute the forward tangent trace  $\frac{\partial y}{\partial x_1}$  for the function we had before

$$y = f(x_1, x_2) = \ln(x_1) + x_1 x_2 - \sin(x_2).$$

- The following table shows both the forward primal trace and the forward tangent trace

Forward primal trace	Forward tangent trace
$v_{-1} = x_1$ $v_0 = x_2$	$\dot{v}_{-1} = \dot{x}_1$ $\dot{v}_0 = \dot{x}_2$
$v_1 = \ln v_{-1}$ $v_2 = v_{-1} \times v_0$ $v_3 = \sin(v_0)$ $v_4 = v_1 + v_2$ $v_5 = v_4 - v_3$	$\dot{v}_1 = \frac{1}{v_{-1}} \dot{v}_{-1}$ $\dot{v}_2 = \dot{v}_{-1} \times v_0 + \dot{v}_0 \times v_{-1}$ $\dot{v}_3 = \dot{v}_0 \times \cos(v_0)$ $\dot{v}_4 = \dot{v}_1 + \dot{v}_2$ $\dot{v}_5 = \dot{v}_4 - \dot{v}_3$
$y = v_5$	$\dot{y} = \dot{v}_5$

## Forward primal trace and tangent trace for $\frac{\partial y}{\partial x_1}$ (II)

We now compute the derivative  $\frac{\partial y}{\partial x_1}$  at  $x_1 = 2, x_2 = 5$ .

## Forward primal trace and tangent trace for $\frac{\partial y}{\partial x_1}$ (II)

We now compute the derivative  $\frac{\partial y}{\partial x_1}$  at  $x_1 = 2, x_2 = 5$ .

Forward primal trace	Forward tangent trace
$v_{-1} = x_1 = 2$	$\dot{v}_{-1} = \dot{x}_1$
$v_0 = x_2 = 5$	$\dot{v}_0 = \dot{x}_2$
$v_1 = \ln v_{-1}$	$\dot{v}_1 = \frac{1}{v_{-1}} \dot{v}_{-1}$
$v_2 = v_{-1} \times v_0$	$\dot{v}_2 = \dot{v}_{-1} \times v_0 + \dot{v}_0 \times v_{-1}$
$v_3 = \sin(v_0)$	$\dot{v}_3 = \dot{v}_0 \times \cos(v_0)$
$v_4 = v_1 + v_2$	$\dot{v}_4 = \dot{v}_1 + \dot{v}_2$
$v_5 = v_4 - v_3$	$\dot{v}_5 = \dot{v}_4 - \dot{v}_3$
$y = v_5$	$\dot{y} = \dot{v}_5$

# Forward primal trace and tangent trace for $\frac{\partial y}{\partial x_1}$ (II)

We now compute the derivative  $\frac{\partial y}{\partial x_1}$  at  $x_1 = 2, x_2 = 5$ .

Forward primal trace	Forward tangent trace
$v_{-1} = x_1 = 2$	$\dot{v}_{-1} = \dot{x}_1 = 1$
$v_0 = x_2 = 5$	$\dot{v}_0 = \dot{x}_2 = 0$
$v_1 = \ln v_{-1}$	$\dot{v}_1 = \frac{1}{v_{-1}} \dot{v}_{-1}$
$v_2 = v_{-1} \times v_0$	$\dot{v}_2 = \dot{v}_{-1} \times v_0 + \dot{v}_0 \times v_{-1}$
$v_3 = \sin(v_0)$	$\dot{v}_3 = \dot{v}_0 \times \cos(v_0)$
$v_4 = v_1 + v_2$	$\dot{v}_4 = \dot{v}_1 + \dot{v}_2$
$v_5 = v_4 - v_3$	$\dot{v}_5 = \dot{v}_4 - \dot{v}_3$
$y = v_5$	$\dot{y} = \dot{v}_5$

## Forward primal trace and tangent trace for $\frac{\partial y}{\partial x_1}$ (II)

We now compute the derivative  $\frac{\partial y}{\partial x_1}$  at  $x_1 = 2, x_2 = 5$ .

Forward primal trace	Forward tangent trace
$v_{-1} = x_1 = 2$	$\dot{v}_{-1} = \dot{x}_1 = 1$
$v_0 = x_2 = 5$	$\dot{v}_0 = \dot{x}_2 = 0$
$v_1 = \ln v_{-1} = \ln 2$	$\dot{v}_1 = \frac{1}{v_{-1}} \dot{v}_{-1} = \frac{1}{2}(1)$
$v_2 = v_{-1} \times v_0$	$\dot{v}_2 = \dot{v}_{-1} \times v_0 + \dot{v}_0 \times v_{-1}$
$v_3 = \sin(v_0)$	$\dot{v}_3 = \dot{v}_0 \times \cos(v_0)$
$v_4 = v_1 + v_2$	$\dot{v}_4 = \dot{v}_1 + \dot{v}_2$
$v_5 = v_4 - v_3$	$\dot{v}_5 = \dot{v}_4 - \dot{v}_3$
$y = v_5$	$\dot{y} = \dot{v}_5$

## Forward primal trace and tangent trace for $\frac{\partial y}{\partial x_1}$ (II)

We now compute the derivative  $\frac{\partial y}{\partial x_1}$  at  $x_1 = 2, x_2 = 5$ .

Forward primal trace	Forward tangent trace
$v_{-1} = x_1 = 2$	$\dot{v}_{-1} = \dot{x}_1 = 1$
$v_0 = x_2 = 5$	$\dot{v}_0 = \dot{x}_2 = 0$
$v_1 = \ln v_{-1} = \ln 2$	$\dot{v}_1 = \frac{1}{v_{-1}} \dot{v}_{-1} = \frac{1}{2}(1)$
$v_2 = v_{-1} \times v_0 = 2 \times 5$	$\dot{v}_2 = \dot{v}_{-1} \times v_0 + \dot{v}_0 \times v_{-1} = 1 \times 5 + 0 \times 2$
$v_3 = \sin(v_0)$	$\dot{v}_3 = \dot{v}_0 \times \cos(v_0)$
$v_4 = v_1 + v_2$	$\dot{v}_4 = \dot{v}_1 + \dot{v}_2$
$v_5 = v_4 - v_3$	$\dot{v}_5 = \dot{v}_4 - \dot{v}_3$
$y = v_5$	$\dot{y} = \dot{v}_5$



## Forward primal trace and tangent trace for $\frac{\partial y}{\partial x_1}$ (II)

We now compute the derivative  $\frac{\partial y}{\partial x_1}$  at  $x_1 = 2, x_2 = 5$ .

Forward primal trace	Forward tangent trace
$v_{-1} = x_1 = 2$	$\dot{v}_{-1} = \dot{x}_1 = 1$
$v_0 = x_2 = 5$	$\dot{v}_0 = \dot{x}_2 = 0$
$v_1 = \ln v_{-1} = \ln 2$	$\dot{v}_1 = \frac{1}{v_{-1}} \dot{v}_{-1} = \frac{1}{2}(1)$
$v_2 = v_{-1} \times v_0 = 2 \times 5$	$\dot{v}_2 = \dot{v}_{-1} \times v_0 + \dot{v}_0 \times v_{-1} = 1 \times 5 + 0 \times 2$
$v_3 = \sin(v_0) = \sin(5)$	$\dot{v}_3 = \dot{v}_0 \times \cos(v_0) = 0 \times \cos(5)$
$v_4 = v_1 + v_2$	$\dot{v}_4 = \dot{v}_1 + \dot{v}_2$
$v_5 = v_4 - v_3$	$\dot{v}_5 = \dot{v}_4 - \dot{v}_3$
$y = v_5$	$\dot{y} = \dot{v}_5$

## Forward primal trace and tangent trace for $\frac{\partial y}{\partial x_1}$ (II)

We now compute the derivative  $\frac{\partial y}{\partial x_1}$  at  $x_1 = 2, x_2 = 5$ .

Forward primal trace	Forward tangent trace
$v_{-1} = x_1 = 2$	$\dot{v}_{-1} = \dot{x}_1 = 1$
$v_0 = x_2 = 5$	$\dot{v}_0 = \dot{x}_2 = 0$
$v_1 = \ln v_{-1} = \ln 2$	$\dot{v}_1 = \frac{1}{v_{-1}} \dot{v}_{-1} = \frac{1}{2}(1)$
$v_2 = v_{-1} \times v_0 = 2 \times 5$	$\dot{v}_2 = \dot{v}_{-1} \times v_0 + \dot{v}_0 \times v_{-1} = 1 \times 5 + 0 \times 2$
$v_3 = \sin(v_0) = \sin(5)$	$\dot{v}_3 = \dot{v}_0 \times \cos(v_0) = 0 \times \cos(5)$
$v_4 = v_1 + v_2 = 0.693 + 10$	$\dot{v}_4 = \dot{v}_1 + \dot{v}_2 = 0.5 + 5$
$v_5 = v_4 - v_3$	$\dot{v}_5 = \dot{v}_4 - \dot{v}_3$
$y = v_5$	$\dot{y} = \dot{v}_5$

# Forward primal trace and tangent trace for $\frac{\partial y}{\partial x_1}$ (II)

We now compute the derivative  $\frac{\partial y}{\partial x_1}$  at  $x_1 = 2, x_2 = 5$ .

Forward primal trace	Forward tangent trace
$v_{-1} = x_1 = 2$	$\dot{v}_{-1} = \dot{x}_1 = 1$
$v_0 = x_2 = 5$	$\dot{v}_0 = \dot{x}_2 = 0$
$v_1 = \ln v_{-1} = \ln 2$	$\dot{v}_1 = \frac{1}{v_{-1}} \dot{v}_{-1} = \frac{1}{2}(1)$
$v_2 = v_{-1} \times v_0 = 2 \times 5$	$\dot{v}_2 = \dot{v}_{-1} \times v_0 + \dot{v}_0 \times v_{-1} = 1 \times 5 + 0 \times 2$
$v_3 = \sin(v_0) = \sin(5)$	$\dot{v}_3 = \dot{v}_0 \times \cos(v_0) = 0 \times \cos(5)$
$v_4 = v_1 + v_2 = 0.693 + 10$	$\dot{v}_4 = \dot{v}_1 + \dot{v}_2 = 0.5 + 5$
$v_5 = v_4 - v_3 = 10.693 + 0.959$	$\dot{v}_5 = \dot{v}_4 - \dot{v}_3 = 5.5 - 0$
$y = v_5$	$\dot{y} = \dot{v}_5$

## Forward primal trace and tangent trace for $\frac{\partial y}{\partial x_1}$ (II)

We now compute the derivative  $\frac{\partial y}{\partial x_1}$  at  $x_1 = 2, x_2 = 5$ .

Forward primal trace	Forward tangent trace
$v_{-1} = x_1 = 2$	$\dot{v}_{-1} = \dot{x}_1 = 1$
$v_0 = x_2 = 5$	$\dot{v}_0 = \dot{x}_2 = 0$
$v_1 = \ln v_{-1} = \ln 2$	$\dot{v}_1 = \frac{1}{v_{-1}} \dot{v}_{-1} = \frac{1}{2}(1)$
$v_2 = v_{-1} \times v_0 = 2 \times 5$	$\dot{v}_2 = \dot{v}_{-1} \times v_0 + \dot{v}_0 \times v_{-1} = 1 \times 5 + 0 \times 2$
$v_3 = \sin(v_0) = \sin(5)$	$\dot{v}_3 = \dot{v}_0 \times \cos(v_0) = 0 \times \cos(5)$
$v_4 = v_1 + v_2 = 0.693 + 10$	$\dot{v}_4 = \dot{v}_1 + \dot{v}_2 = 0.5 + 5$
$v_5 = v_4 - v_3 = 10.693 + 0.959$	$\dot{v}_5 = \dot{v}_4 - \dot{v}_3 = 5.5 - 0$
$y = v_5 = 11.652$	$\dot{y} = \dot{v}_5 = 5.5$

## Forward primal trace and tangent trace for $\frac{\partial y}{\partial x_1}$ (II)

We now compute the derivative  $\frac{\partial y}{\partial x_1}$  at  $x_1 = 2, x_2 = 5$ .

Forward primal trace	Forward tangent trace
$v_{-1} = x_1 = 2$	$\dot{v}_{-1} = \dot{x}_1 = 1$
$v_0 = x_2 = 5$	$\dot{v}_0 = \dot{x}_2 = 0$
$v_1 = \ln v_{-1} = \ln 2$	$\dot{v}_1 = \frac{1}{v_{-1}} \dot{v}_{-1} = \frac{1}{2}(1)$
$v_2 = v_{-1} \times v_0 = 2 \times 5$	$\dot{v}_2 = \dot{v}_{-1} \times v_0 + \dot{v}_0 \times v_{-1} = 1 \times 5 + 0 \times 2$
$v_3 = \sin(v_0) = \sin(5)$	$\dot{v}_3 = \dot{v}_0 \times \cos(v_0) = 0 \times \cos(5)$
$v_4 = v_1 + v_2 = 0.693 + 10$	$\dot{v}_4 = \dot{v}_1 + \dot{v}_2 = 0.5 + 5$
$v_5 = v_4 - v_3 = 10.693 + 0.959$	$\dot{v}_5 = \dot{v}_4 - \dot{v}_3 = 5.5 - 0$
$y = v_5 = 11.652$	$\dot{y} = \dot{v}_5 = 5.5$

If we want to compute  $\frac{\partial y}{\partial x_2}$  instead, we set  $\dot{v}_{-1} = 0$  and  $\dot{v}_0 = 1$ .

# Generalisation to the Jacobian of a function

- Let  $\mathbf{f} : \mathbb{R}^n \rightarrow \mathbb{R}^m$  be a function with  $n$  independent variables  $x_i$  and  $m$  dependent variables  $y_j$ .
- The derivatives in the Jacobian,  $\frac{\partial y_j}{\partial x_i}$ , are computed by making  $\dot{x}_i = 1$  initially in the forward pass and all the other derivatives  $\dot{x}_k = 0$  for  $k \neq i$ .
- The values of the derivatives at  $\mathbf{x} = \mathbf{a}$ ,

$$\dot{y}_j = \left. \frac{\partial y_j}{\partial x_i} \right|_{\mathbf{x}=\mathbf{a}}.$$

are obtained by a forward pass of AD.

- Notice that for a specific  $x_i$ , we can compute all the derivatives  $\frac{\partial y_j}{\partial x_i}$  for  $j = 1, \dots, m$ , which corresponds to the column  $i$  in the Jacobian.
- To compute the whole Jacobian, we need  $n$  forward passes, one per input variable.

# Complexity

- AD with forward mode is efficient for functions like  $\mathbf{f} : \mathbb{R} \rightarrow \mathbb{R}^m$ .
- The reason, as we saw before, is because we can compute all the derivatives  $\frac{\partial y_j}{\partial x}$  for  $j = 1, \dots, m$  in one pass.
- In the other extreme,  $f : \mathbb{R}^n \rightarrow \mathbb{R}$ , it needs  $n$  forward passes and it can become computationally expensive when  $n$  is large.
- In general, when  $n \gg m$ , the reverse mode of AD is preferred.

# Contents

Derivatives and ways to compute them

AD modes

Forward mode

Reverse mode

Implementations



# Backpropagate

- AD in reverse mode propagates derivatives backwards from a given output.
- It is done by computing intermediate variables for  $v_i$  known as *adjoints*,

$$\bar{v}_i = \frac{\partial y_j}{\partial v_i},$$

representing the sensitivity of output  $y_j$  to input  $v_i$ .

- AD in reverse mode uses two-phases
  - a *forward* step to compute the variables  $v_i$  and to book-keep dependencies in the computational graph.
  - a *backward* or *reverse* step, in which the adjoints are used to compute the derivatives, starting from the outputs and going back to the inputs.

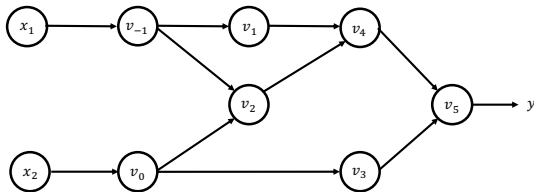
## Example (I)

- Let us go back to the example we saw before

$$y = f(x_1, x_2) = \ln(x_1) + x_1 x_2 - \sin(x_2),$$

and focus on  $v_0$  ( $v_0 = x_2$ ).

- We want to compute the adjoint  $\bar{v}_0 = \frac{\partial y}{\partial v_0}$ , this is, how the change in  $v_0$  affects the output  $y$ .
- From the computational graph, we see that  $v_0$  affects  $y$  through  $v_2$  and  $v_3$ ,



## Example (II)

- So the contribution of  $v_0$  to  $y$  is given as

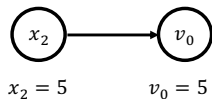
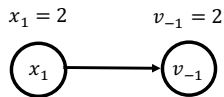
$$\frac{\partial y}{\partial v_0} = \frac{\partial y}{\partial v_2} \frac{\partial v_2}{\partial v_0} + \frac{\partial y}{\partial v_3} \frac{\partial v_3}{\partial v_0}.$$

- By definition  $\frac{\partial y}{\partial v_2} = \bar{v}_2$  and  $\frac{\partial y}{\partial v_3} = \bar{v}_3$ , so we can write the expression above as

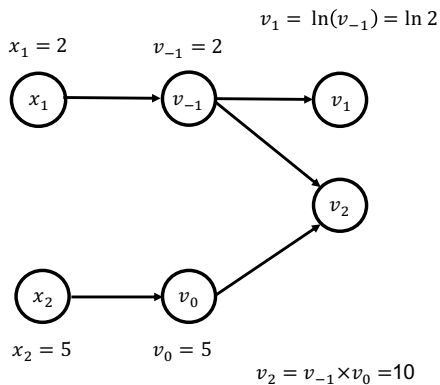
$$\bar{v}_0 = \bar{v}_2 \frac{\partial v_2}{\partial v_0} + \bar{v}_3 \frac{\partial v_3}{\partial v_0}.$$

- After the forward pass to compute  $v_i$ , the reverse pass computes the adjoints, starting with  $\bar{v}_5 = \bar{y} = \frac{\partial y}{\partial y} = 1$ , and computing  $\frac{\partial y}{\partial x_1} = \bar{x}_1$  and  $\frac{\partial y}{\partial x_2} = \bar{x}_2$  at the end.

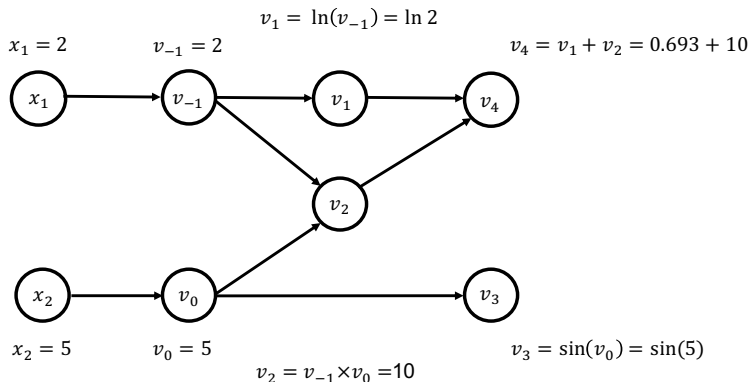
# Forward primal trace (forward pass)



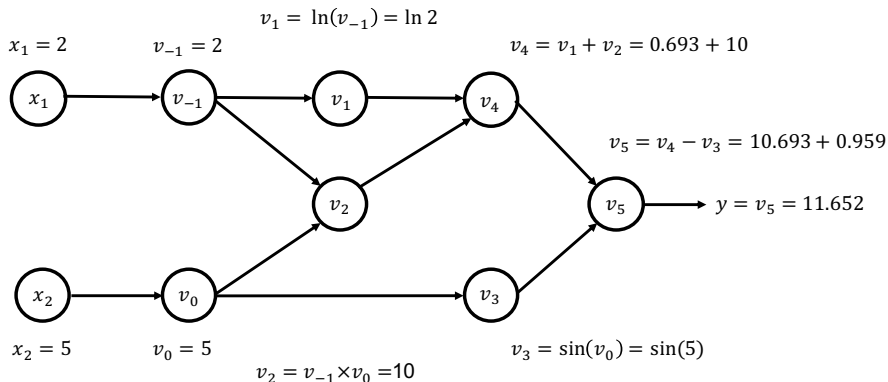
# Forward primal trace (forward pass)



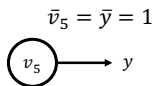
# Forward primal trace (forward pass)



# Forward primal trace (forward pass)

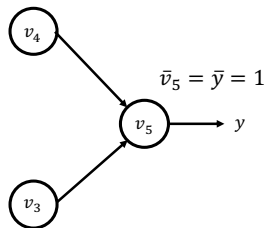


# Reverse adjoint (derivative) trace (reverse pass)

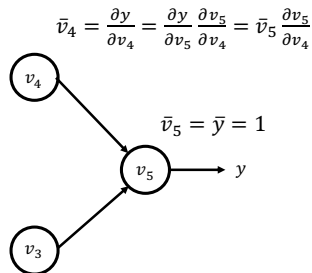




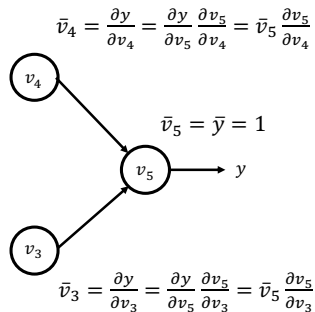
# Reverse adjoint (derivative) trace (reverse pass)



# Reverse adjoint (derivative) trace (reverse pass)

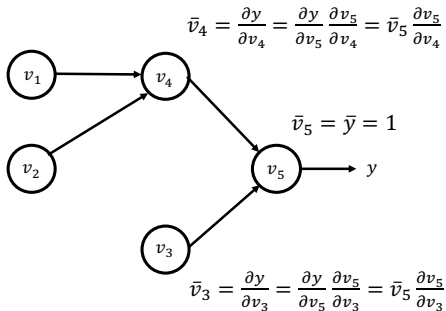


# Reverse adjoint (derivative) trace (reverse pass)



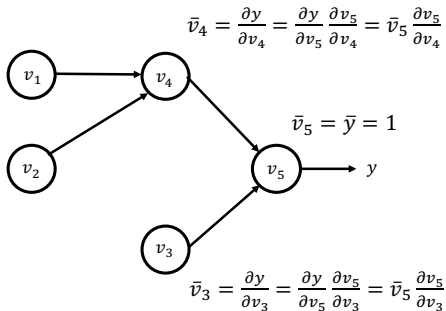
# Reverse adjoint (derivative) trace (reverse pass)

$$\bar{v}_1 = \frac{\partial y}{\partial v_1} = \frac{\partial y}{\partial v_5} \frac{\partial v_5}{\partial v_4} \frac{\partial v_4}{\partial v_1} = \bar{v}_4 \frac{\partial v_4}{\partial v_1}$$



# Reverse adjoint (derivative) trace (reverse pass)

$$\bar{v}_1 = \frac{\partial y}{\partial v_1} = \frac{\partial y}{\partial v_5} \frac{\partial v_5}{\partial v_4} \frac{\partial v_4}{\partial v_1} = \bar{v}_4 \frac{\partial v_4}{\partial v_1}$$



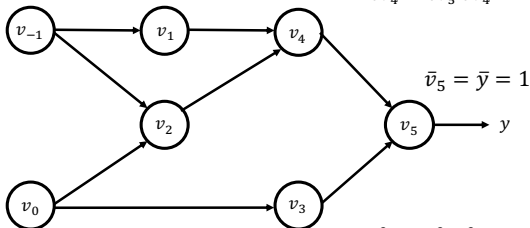
$$\bar{v}_2 = \frac{\partial y}{\partial v_2} = \frac{\partial y}{\partial v_5} \frac{\partial v_5}{\partial v_4} \frac{\partial v_4}{\partial v_2} = \bar{v}_4 \frac{\partial v_4}{\partial v_2}$$

# Reverse adjoint (derivative) trace (reverse pass)

$$\bar{v}_1 = \frac{\partial y}{\partial v_1} = \frac{\partial y}{\partial v_5} \frac{\partial v_5}{\partial v_4} \frac{\partial v_4}{\partial v_1} = \bar{v}_4 \frac{\partial v_4}{\partial v_1}$$

$$\bar{v}_{-1} = \frac{\partial y}{\partial v_{-1}} = \frac{\partial y}{\partial v_1} \frac{\partial v_1}{\partial v_{-1}} + \frac{\partial y}{\partial v_2} \frac{\partial v_2}{\partial v_{-1}} = \bar{v}_1 \frac{\partial v_1}{\partial v_{-1}} + \bar{v}_2 \frac{\partial v_2}{\partial v_{-1}}$$

$$\bar{v}_4 = \frac{\partial y}{\partial v_4} = \frac{\partial y}{\partial v_5} \frac{\partial v_5}{\partial v_4} = \bar{v}_5 \frac{\partial v_5}{\partial v_4}$$



$$\bar{v}_3 = \frac{\partial y}{\partial v_3} = \frac{\partial y}{\partial v_5} \frac{\partial v_5}{\partial v_3} = \bar{v}_5 \frac{\partial v_5}{\partial v_3}$$

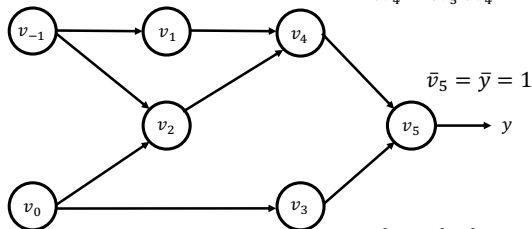
$$\bar{v}_2 = \frac{\partial y}{\partial v_2} = \frac{\partial y}{\partial v_5} \frac{\partial v_5}{\partial v_4} \frac{\partial v_4}{\partial v_2} = \bar{v}_4 \frac{\partial v_4}{\partial v_2}$$

# Reverse adjoint (derivative) trace (reverse pass)

$$\bar{v}_1 = \frac{\partial y}{\partial v_1} = \frac{\partial y}{\partial v_5} \frac{\partial v_5}{\partial v_4} \frac{\partial v_4}{\partial v_1} = \bar{v}_4 \frac{\partial v_4}{\partial v_1}$$

$$\bar{v}_{-1} = \frac{\partial y}{\partial v_{-1}} = \frac{\partial y}{\partial v_1} \frac{\partial v_1}{\partial v_{-1}} + \frac{\partial y}{\partial v_2} \frac{\partial v_2}{\partial v_{-1}} = \bar{v}_1 \frac{\partial v_1}{\partial v_{-1}} + \bar{v}_2 \frac{\partial v_2}{\partial v_{-1}}$$

$$\bar{v}_4 = \frac{\partial y}{\partial v_4} = \frac{\partial y}{\partial v_5} \frac{\partial v_5}{\partial v_4} = \bar{v}_5 \frac{\partial v_5}{\partial v_4}$$



$$\bar{v}_0 = \frac{\partial y}{\partial v_0} = \frac{\partial y}{\partial v_2} \frac{\partial v_2}{\partial v_0} + \frac{\partial y}{\partial v_3} \frac{\partial v_3}{\partial v_0} = \bar{v}_2 \frac{\partial v_2}{\partial v_0} + \bar{v}_3 \frac{\partial v_3}{\partial v_0}$$

$$\bar{v}_3 = \frac{\partial y}{\partial v_3} = \frac{\partial y}{\partial v_5} \frac{\partial v_5}{\partial v_3} = \bar{v}_5 \frac{\partial v_5}{\partial v_3}$$

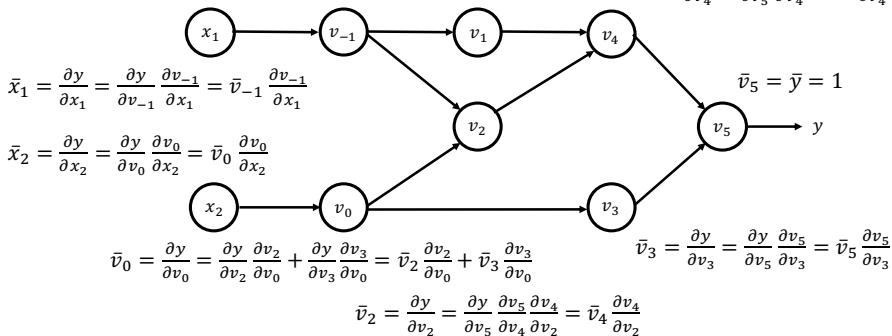
$$\bar{v}_2 = \frac{\partial y}{\partial v_2} = \frac{\partial y}{\partial v_5} \frac{\partial v_5}{\partial v_4} \frac{\partial v_4}{\partial v_2} = \bar{v}_4 \frac{\partial v_4}{\partial v_2}$$

# Reverse adjoint (derivative) trace (reverse pass)

$$\bar{v}_1 = \frac{\partial y}{\partial v_1} = \frac{\partial y}{\partial v_5} \frac{\partial v_5}{\partial v_4} \frac{\partial v_4}{\partial v_1} = \bar{v}_4 \frac{\partial v_4}{\partial v_1}$$

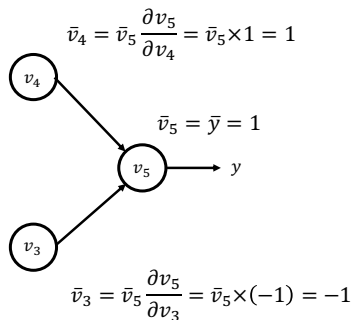
$$\bar{v}_{-1} = \frac{\partial y}{\partial v_{-1}} = \frac{\partial y}{\partial v_1} \frac{\partial v_1}{\partial v_{-1}} + \frac{\partial y}{\partial v_2} \frac{\partial v_2}{\partial v_{-1}} = \bar{v}_1 \frac{\partial v_1}{\partial v_{-1}} + \bar{v}_2 \frac{\partial v_2}{\partial v_{-1}}$$

$$\bar{v}_4 = \frac{\partial y}{\partial v_4} = \frac{\partial y}{\partial v_5} \frac{\partial v_5}{\partial v_4} = \bar{v}_5 \frac{\partial v_5}{\partial v_4}$$



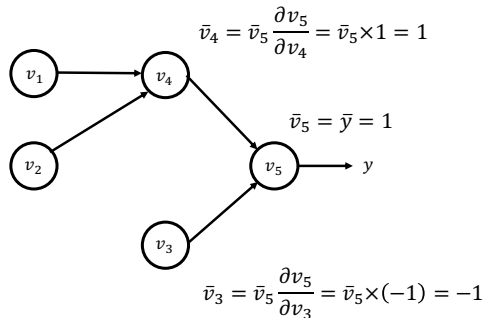


# Numerical evaluation of the reverse adjoint trace



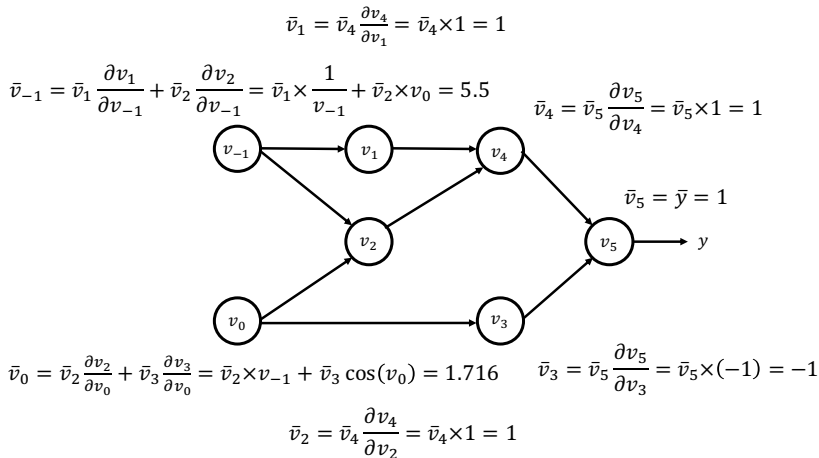
# Numerical evaluation of the reverse adjoint trace

$$\bar{v}_1 = \bar{v}_4 \frac{\partial v_4}{\partial v_1} = \bar{v}_4 \times 1 = 1$$



$$\bar{v}_2 = \bar{v}_4 \frac{\partial v_4}{\partial v_2} = \bar{v}_4 \times 1 = 1$$

# Numerical evaluation of the reverse adjoint trace



# Numerical evaluation of the reverse adjoint trace

$$\bar{v}_1 = \bar{v}_4 \frac{\partial v_4}{\partial v_1} = \bar{v}_4 \times 1 = 1$$

$$\bar{v}_{-1} = \bar{v}_1 \frac{\partial v_1}{\partial v_{-1}} + \bar{v}_2 \frac{\partial v_2}{\partial v_{-1}} = \bar{v}_1 \times \frac{1}{v_{-1}} + \bar{v}_2 \times v_0 = 5.5$$

$$\bar{v}_4 = \bar{v}_5 \frac{\partial v_5}{\partial v_4} = \bar{v}_5 \times 1 = 1$$

$$\bar{x}_1 = \bar{v}_{-1} \frac{\partial v_{-1}}{\partial x_1} = 5.5 \times 1 = 5.5$$

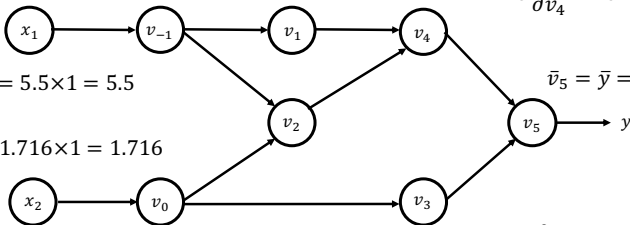
$$\bar{x}_2 = \bar{v}_0 \frac{\partial v_0}{\partial x_2} = 1.716 \times 1 = 1.716$$

$$\bar{v}_5 = \bar{y} = 1$$

$$\bar{v}_0 = \bar{v}_2 \frac{\partial v_2}{\partial v_0} + \bar{v}_3 \frac{\partial v_3}{\partial v_0} = \bar{v}_2 \times v_{-1} + \bar{v}_3 \cos(v_0) = 1.716$$

$$\bar{v}_3 = \bar{v}_5 \frac{\partial v_5}{\partial v_3} = \bar{v}_5 \times (-1) = -1$$

$$\bar{v}_2 = \bar{v}_4 \frac{\partial v_4}{\partial v_2} = \bar{v}_4 \times 1 = 1$$



# Complexity

- Reverse mode AD performs better when  $n \gg m$ .
- The downside is the cost of increased storage, since we need to save intermediate values for  $v_i$  in the evaluation trace.

# Reverse mode AD and backpropagation

- ❑ Reverse mode AD is the algorithm used to train neural networks and deep learning models.
- ❑ To train a neural network model, we optimise an objective function,  $E(\mathbf{w}) : \mathbb{R}^n \rightarrow \mathbb{R}^m$  that usually depends on a high-dimensional input vector of parameters  $\mathbf{w} \in \mathbb{R}^n$ , with  $n \gg m$ .
- ❑ In the machine learning community, reverse mode AD goes by the name of *backpropagation*, which you will see again in the session on neural networks.

# Contents

Derivatives and ways to compute them

AD modes

Forward mode

Reverse mode

Implementations

# AD implementations

Table 5: Survey of AD implementations. Tools developed primarily for machine learning are highlighted in bold.

Language	Tool	Type	Mode	Institution / Project	Reference	URL
AMPL	AMPL	INT	F, R	Bell Laboratories	Fourer et al. (2002)	<a href="http://www.ampl.com/">http://www.ampl.com/</a>
C, C++	ADIC	ST	F, R	Argonne National Laboratory	Bischof et al. (1997)	<a href="http://www.mcs.anl.gov/research/projects/adic/">http://www.mcs.anl.gov/research/projects/adic/</a>
	ADOL-C	OO	F, R	Computational Infrastructure for Operations Research	Walther and Griewank (2012)	<a href="https://projects.coin-or.org/ADOL-C">https://projects.coin-or.org/ADOL-C</a>
C++	Ceres Solver	LIB	F	Google		<a href="http://ceres-solver.org/">http://ceres-solver.org/</a>
	CppAD	OO	F, R	Computational Infrastructure for Operations Research	Bell and Burke (2008)	<a href="http://www.coin-or.org/CppAD/">http://www.coin-or.org/CppAD/</a>
	FADBAD++	OO	F, R	Technical University of Denmark	Bendtsen and Stauning (1996)	<a href="http://www.fadbad.com/fadbad.html">http://www.fadbad.com/fadbad.html</a>
	Mxzyptik	OO	F	Fermi National Accelerator Laboratory	Ostiguy and Michelotti (2007)	
C#	AutoDiff	LIB	R	George Mason Univ., Dept. of Computer Science	Shtof et al. (2013)	<a href="http://autodiff.codeplex.com/">http://autodiff.codeplex.com/</a>
F#, C#	DiffSharp	OO	F, R	Maynooth University, Microsoft Research Cambridge	Baydin et al. (2016a)	<a href="http://diffsharp.github.io">http://diffsharp.github.io</a>
Fortran	ADIFOR	ST	F, R	Argonne National Laboratory	Bischof et al. (1996)	<a href="http://www.mcs.anl.gov/research/projects/adifor/">http://www.mcs.anl.gov/research/projects/adifor/</a>
	NAGWare	COM	F, R	Numerical Algorithms Group	Naumann and Riehme (2005)	<a href="http://www.nag.co.uk/nagware/Research/ad_overview.asp">http://www.nag.co.uk/nagware/Research/ad_overview.asp</a>
	TAMC	ST	R	Max Planck Institute for Meteorology	Giering and Kaminski (1998)	<a href="http://autodiff.com/tamc/">http://autodiff.com/tamc/</a>
Fortran, C	COSY	INT	F	Michigan State Univ., Biomedical and Physical Sci.	Berz et al. (1996)	<a href="http://www.bt.pa.msu.edu/index_cosy.htm">http://www.bt.pa.msu.edu/index_cosy.htm</a>
	Tapenade	ST	F, R	INRIA Sophia-Antipolis	Hascoët and Pascual (2013)	<a href="http://www-sop.inria.fr/tropics/tapenade.html">http://www-sop.inria.fr/tropics/tapenade.html</a>
Haskell	ad	OO	F, R	Haskell package		<a href="http://hackage.haskell.org/package/ad">http://hackage.haskell.org/package/ad</a>
Java	ADiJaC	ST	F, R	University Politehnica of Bucharest	Slusanschi and Dumitrel (2016)	<a href="http://adijac.cs.pub.ro">http://adijac.cs.pub.ro</a>
	Deriva	LIB	R	Java & Clojure library		<a href="https://github.com/lanber/Deriva">https://github.com/lanber/Deriva</a>
Julia	JuliaDiff	OO	F, R	Julia packages	Revels et al. (2016a)	<a href="http://www.juliadiff.org/">http://www.juliadiff.org/</a>
Lua	torch-autograd	OO	R	Twitter Cortex		<a href="https://github.com/twitter/torch-autograd">https://github.com/twitter/torch-autograd</a>
MATLAB	ADiMat	ST	F, R	Technical University of Darmstadt, Scientific Comp.	Willkomm and Vehrenschild (2013)	<a href="http://adimat.sc.informatik.tu-darmstadt.de/">http://adimat.sc.informatik.tu-darmstadt.de/</a>
	INTLab	OO	F	Hamburg Univ. of Technology, Inst. for Reliable Comp.	Rump (1999)	<a href="http://www.ti3.tu-harburg.de/rump/intlab/">http://www.ti3.tu-harburg.de/rump/intlab/</a>
	TOMLAB/MAD	OO	F	Cranfield University & Tomlab Optimization Inc.	Forth (2006)	<a href="http://tomlab.biz/products/mad">http://tomlab.biz/products/mad</a>
Python	ad	OO	R	Python package		<a href="https://pypi.python.org/pypi/ad">https://pypi.python.org/pypi/ad</a>
	autograd	OO	F, R	Harvard Intelligent Probabilistic Systems Group	Maclaurin (2016)	<a href="https://github.com/HIPS/autograd">https://github.com/HIPS/autograd</a>
	Chainer	OO	R	Preferred Networks	Tokui et al. (2015)	<a href="https://chainer.org/">https://chainer.org/</a>
	PyTorch	OO	R	PyTorch core team	Paszke et al. (2017)	<a href="http://pytorch.org/">http://pytorch.org/</a>
	Tangent	ST	F, R	Google Brain	van Merriënboer et al. (2017)	<a href="https://github.com/google/tangent">https://github.com/google/tangent</a>
Scheme	R6RS-AD	OO	F, R	Purdue Univ., School of Electrical and Computer Eng.		<a href="https://github.com/qobi/R6RS-AD">https://github.com/qobi/R6RS-AD</a>
	Scmutils	OO	F	MIT Computer Science and Artificial Intelligence Lab.	Sussman and Wisdom (2001)	<a href="http://groups.csail.mit.edu/mac/users/gjs/6946/refman.txt">http://groups.csail.mit.edu/mac/users/gjs/6946/refman.txt</a>
	Stalingrad	COM	F, R	Purdue Univ., School of Electrical and Computer Eng.	Pearlmutter and Siskind (2008)	<a href="http://www.bcl.hamilton.ie/~qobi/stalingrad/">http://www.bcl.hamilton.ie/~qobi/stalingrad/</a>

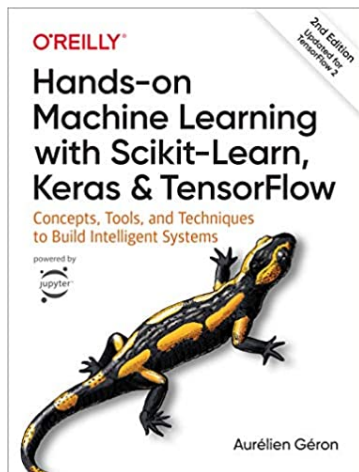
F: Forward, R: Reverse; COM: Compiler, INT: Interpreter, LIB: Library, OO: Operator overloading, ST: Source transformation



# Two popular implementations in the ML community



# References



Appendix D of “Hands-On Machine Learning with Scikit-Learn, Keras and TensorFlow”

## Automatic Differentiation in Machine Learning: a Survey

**Atılım Güneş Baydin**

*Department of Engineering Science  
University of Oxford  
Oxford OX1 3PJ, United Kingdom*

GUNES@ROBOTS.OX.AC.UK

**Barak A. Pearlmutter**

*Department of Computer Science  
National University of Ireland Maynooth  
Maynooth, Co. Kildare, Ireland*

BARAK@PEARLMUTTER.NET

**Alexey Andreyevich Radul**

*Department of Brain and Cognitive Sciences  
Massachusetts Institute of Technology  
Cambridge, MA 02139, United States*

AXCH@MIT.EDU

**Jeffrey Mark Siskind**

*School of Electrical and Computer Engineering  
Purdue University  
West Lafayette, IN 47907, United States*

QOBI@PURDUE.EDU