Computational Graphs

A ubiquitous problem in Machine Learning

- You have some function f which depends on some parameters θ : $f(x \mid \theta)$
- You have some example input-output pairs $(x_0, y_0), (x_1, y_1), \ldots (x_n, y_n)$
- For each input-output pair you can compute an error function L between the output of f and the actual y
  - $L(y, \overline{y}) = L(y_0, f(x_0 \mid \theta))$
- You want to find the value of θ for which the error L is minimum (on average)

$$\min_{(y_i, x_i) \in D} E[\ L(y_i, f(x_i \mid \theta))\ ]$$

A ubiquitous problem in Machine Learning

One way to find the optimal θ is to use the Stochastic Gradient Descent SGD method

- You can compute $\mathbb{E}\left[\frac{dL}{d\theta}\right]$
- $\mathbb{E}\left[\frac{dL}{d\theta}\right]$ has the same dimensionality of θ and it is a vector which points to the direction that maximize L
- The reciprocal $-\mathbb{E}\left[\frac{dL}{d\theta}\right]$ points to the direction that minimize L
- We can take small steps in this direction

$$\theta_{t+1} = \theta_t + \alpha\left(-E\left[\frac{dL}{d\theta_t}\right]\right)$$

[Demo Notebook]

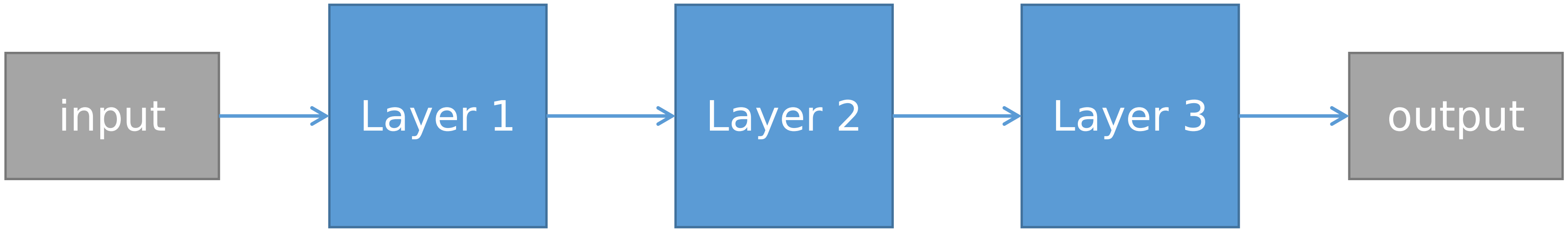
How to efficiently compute $\frac{dL}{d\theta}$

There are several ways to compute $\frac{dL}{d\theta}$

- Manual
  - Not feasible for complex/deep networks
- Symbolic
  - Computationally hard or just plain impossible
- Automatic
  - The go-to solution for machine learning

Chain Rule

A simple Machine Learning model can look like this:



Which can be formalized in:

$$o = l_3(l_2(l_1(i)))$$

This operation is called **composition** can also be written as $l_3 \circ l_2 \circ l_1$

Chain Rule

You may be familiar with this notation:

$$h(x) = f(g(x))$$
$$h'(x) = f'(g(x))g'(x)$$

Unfolding the equation from previous slide $o = l_3(l_2(l_1(i)))$:

$$w_1 = l_1(i)$$
$$w_2 = l_2(w_1)$$
$$o = l_3(w_2)$$

The derivative of o w.r.t. i is then:

$$\frac{do}{di} = \frac{do}{dw_2}\frac{dw_2}{dw_1}\frac{dw_1}{di}$$

The derivative of the composition is the multiplication of the partial derivatives (of the unfolding)

Chain Rule: Example

Given this function:

$$o = sin(x^2)$$

Unfolded:

$$w_1 = x^2$$
$$o = sin(w_1)$$

The derivative $\frac{do}{dx}$ is then:

$$\frac{do}{dx} = \frac{do}{dw_1}\frac{dw_1}{dx}$$

$$\frac{do}{dw_1} = \frac{d(sin(w_1))}{dw_1} = cos(w_1) = cos(x^2)$$

$$\frac{dw_1}{dx} = \frac{d(x^2)}{dx} = 2x$$

Finally:

$$\frac{do}{dx} = \frac{do}{dw_1}\frac{dw_1}{dx} = cos(x^2)2x$$

Chain Rule: Binary Operators

A binary operator is a rule for combining two elements (called operands) to produce another element

$$f\colon A \times B \rightarrow C$$

A binary operator is **closed** if its domain is $A \times A$ and its codomain is A

The closed operator in $\mathbb{R}$ are:

- Addition (+)
- Subtraction (-)
- Multiplication (*)
- Division (/)

Chain Rule: Multiple Variables with Binary Operators

Lets an example of the application of the chain rule with multiple variables and binary operators

$$o = (x + y)sin(x)$$

Lets compute $\frac{do}{dx}$:

$$w_1 = x + y$$
$$w_2 = sin(x)$$
$$o = w_1 w_2$$

$$\frac{dw_1}{dx} = \frac{d(x+y)}{dx} = 1$$

$$\frac{dw_2}{dx} = \frac{d(sin(x))}{dx} = cos(x)$$

$$\frac{do}{dx} = \frac{d(w_1 w_2)}{dx} = w_2 \frac{dw_1}{dx} + w_1 \frac{dw_2}{dx}$$

$$\frac{do}{dx} = w_2 + w_1 cos(x)$$

$$\frac{do}{dx} = sin(x) + (x+y) cos(x)$$

The binary operator caused a split on the derivation flow!

Chain Rule: Binary Operators

Every binary operator cause a **split** in the derivation flow.

In $\mathbb{R}$ **every closed binary operator** create a **summation** of two derivation flows:

$$\frac{d(f(x)\ +\ g(x))}{dx} = \frac{d(f(x))}{dx} + \frac{d(g(x))}{dx}$$

$$\frac{d(f(x)\ g(x))}{dx} = g(x)\frac{d(f(x))}{dx} + f(x)\frac{d(g(x))}{dx}$$

$$\frac{d(f(x)\ -\ g(x))}{dx} = \frac{d(f(x))}{dx} - \frac{d(g(x))}{dx} = \frac{d(f(x))}{dx} + -\frac{d(g(x))}{dx}$$

$$\frac{d(\frac{f(x)}{g(x)})}{dx} = \frac{g(x)\frac{d(f(x))}{dx} - f(x)\frac{d(g(x))}{dx}}{g(x)^2} = \frac{1}{g(x)}\frac{d(f(x))}{dx} + -\frac{f(x)}{g(x)^2}\frac{d(g(x))}{dx}$$

Chain Rule: Binary Operators

In which **each flow** is the **derivative** of one **operand multiplied** for **some value**

$$\frac{d(f(x)\ +\ g(x))}{dx} = 1\frac{d(f(x))}{dx} + 1\frac{d(g(x))}{dx}$$

$$\frac{d(f(x)\ g(x))}{dx} = g(x)\frac{d(f(x))}{dx} + f(x)\frac{d(g(x))}{dx}$$

$$\frac{d(f(x)\ -\ g(x))}{dx} = \frac{d(f(x))}{dx} - \frac{d(g(x))}{dx} = 1\frac{d(f(x))}{dx} + -1\frac{d(g(x))}{dx}$$

$$\frac{d(\frac{f(x)}{g(x)})}{dx} = \frac{g(x)\frac{d(f(x))}{dx} - f(x)\frac{d(g(x))}{dx}}{g(x)^2} = \frac{1}{g(x)}\frac{d(f(x))}{dx} + -\frac{f(x)}{g(x)^2}\frac{d(g(x))}{dx}$$

Computational Graphs

Computational graphs are a way of expressing and evaluating a mathematical expression

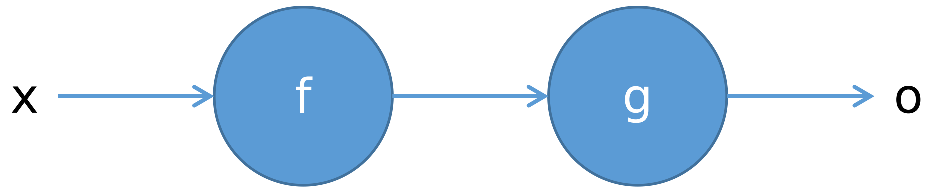
Each operator is represent with a **node**

A node can have **one or more inputs** and **one output**.

Computational Graphs: Composition

A **composition** in a computational graph is a **simple flow**.

For example $o = g(f(x))$:



We can use a Computational Graph to compute the derivatives of an expression

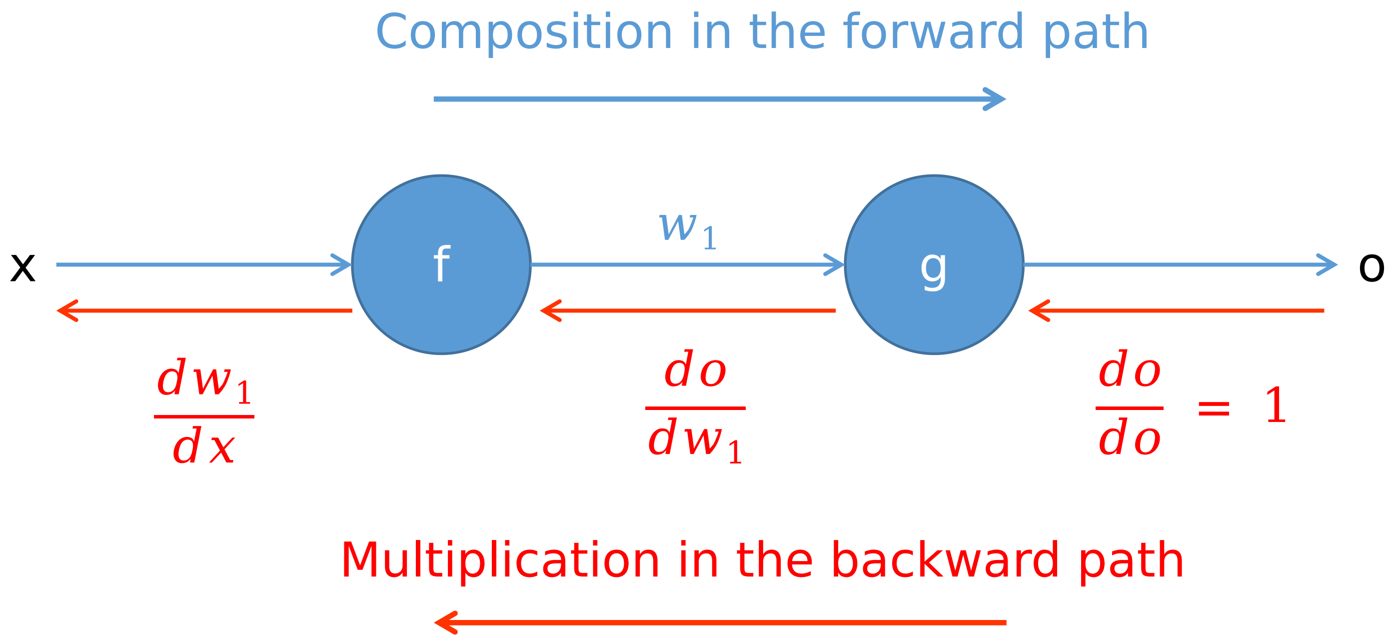
The first method we will see its called **reverse mode differentiation**

Computational Graphs: Composition differentiation

The derivative $\frac{do}{dx}$ of $o = g(f(x))$ where:

$$w_1 = f(x)$$
$$o = g(w_1)$$

$$\frac{do}{dx} = \frac{do}{dw_1}\frac{dw_1}{dx}$$



Computational Graphs: Binary Operator differentiation (dx branch)

The derivative $\frac{do}{dx}$ of $o = f(x)g(y)$ where:

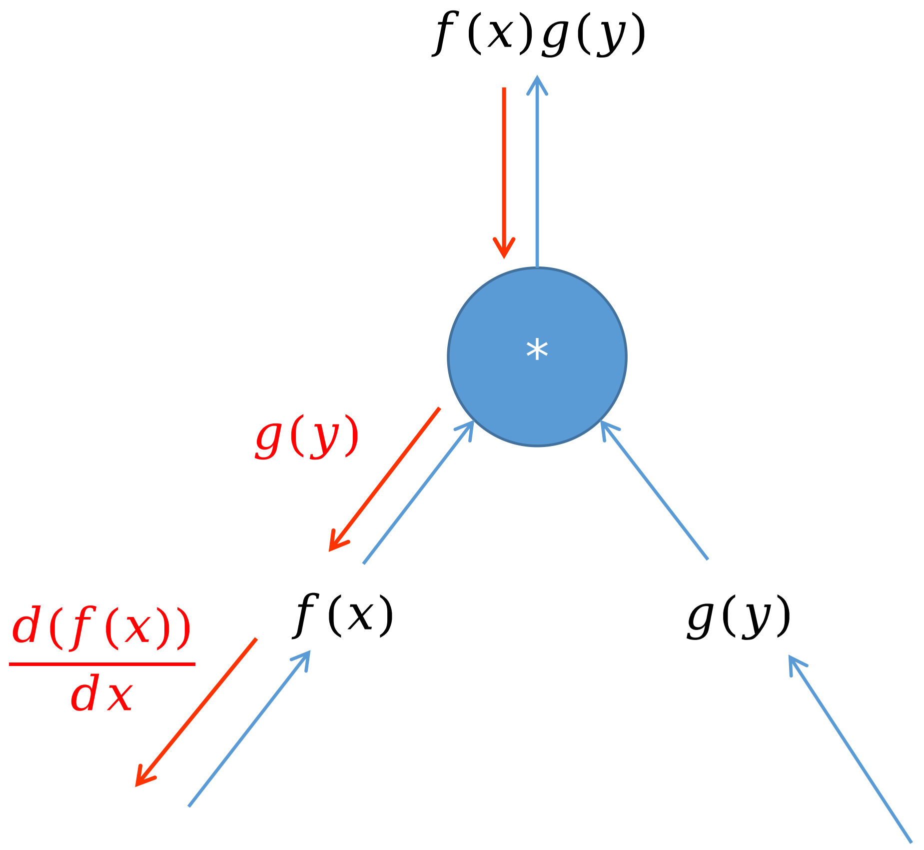
$$
\begin{aligned}
w_1 &= f(x) \\
w_2 &= g(y) \\
o &= w_1 w_2
\end{aligned}
$$

$$\frac{do}{dx} = w_2 \frac{dw_1}{dx} + w_1 \frac{dw_2}{dx} =$$

$$= g(y)\frac{d(f(x))}{dx} + f(x)\frac{d(g(y))}{dx}$$

$$= g(y)\frac{d(f(x))}{dx}$$

This is zero!



Computational Graphs: Binary Operator differentiation (dy branch)

The derivative $\frac{do}{dy}$ of $o = f(x)g(y)$ where:

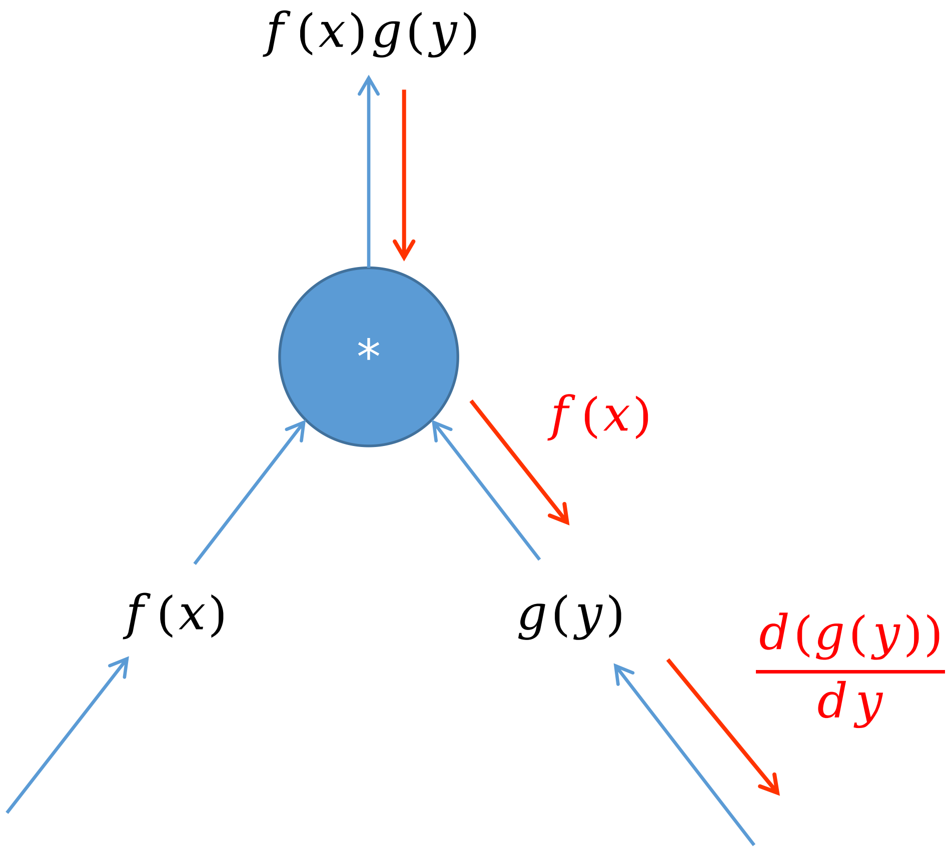
$$w_1 = f(x)$$
$$w_2 = g(y)$$
$$o = w_1 w_2$$

$$\frac{do}{dy} = w_2 \frac{dw_1}{dy} + w_1 \frac{dw_2}{dy} =$$

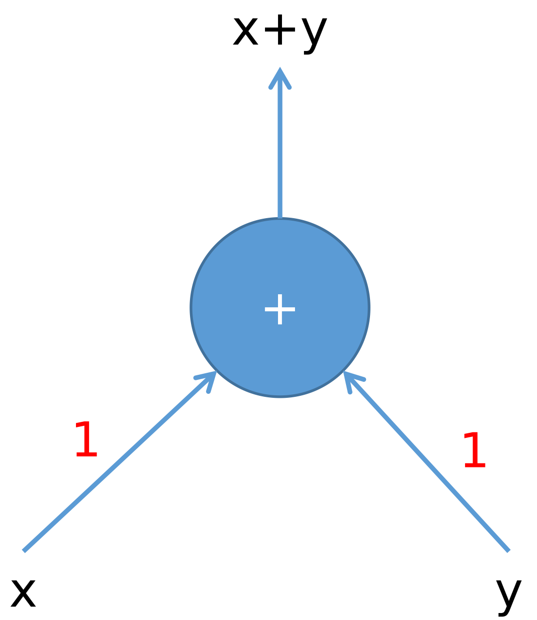
$$= g(y)\frac{d(f(x))}{dy} + f(x)\frac{d(g(y))}{dy}$$

$$= f(x)\frac{d(g(y))}{dy}$$

This is zero!

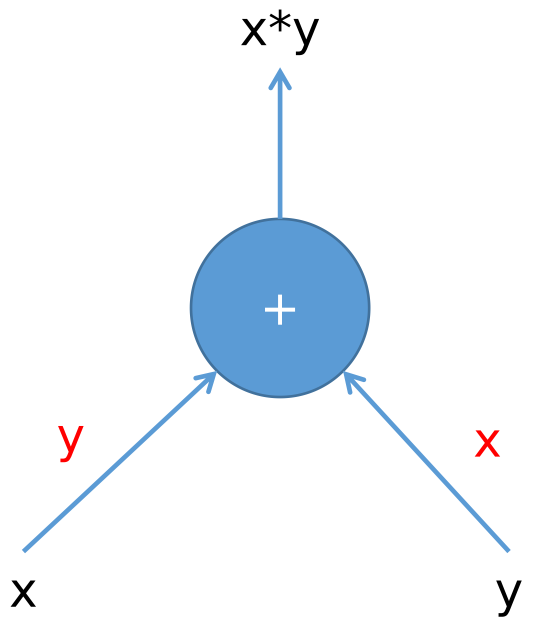


Computational Graphs: Addition and Multiplication nodes



$$\frac{d(x+y)}{dx} = \frac{dx}{dx} + \cancel{\frac{dy}{dx}} = 1$$

$$\frac{d(x+y)}{dy} = \cancel{\frac{dx}{dy}} + \frac{dy}{dy} = 1$$

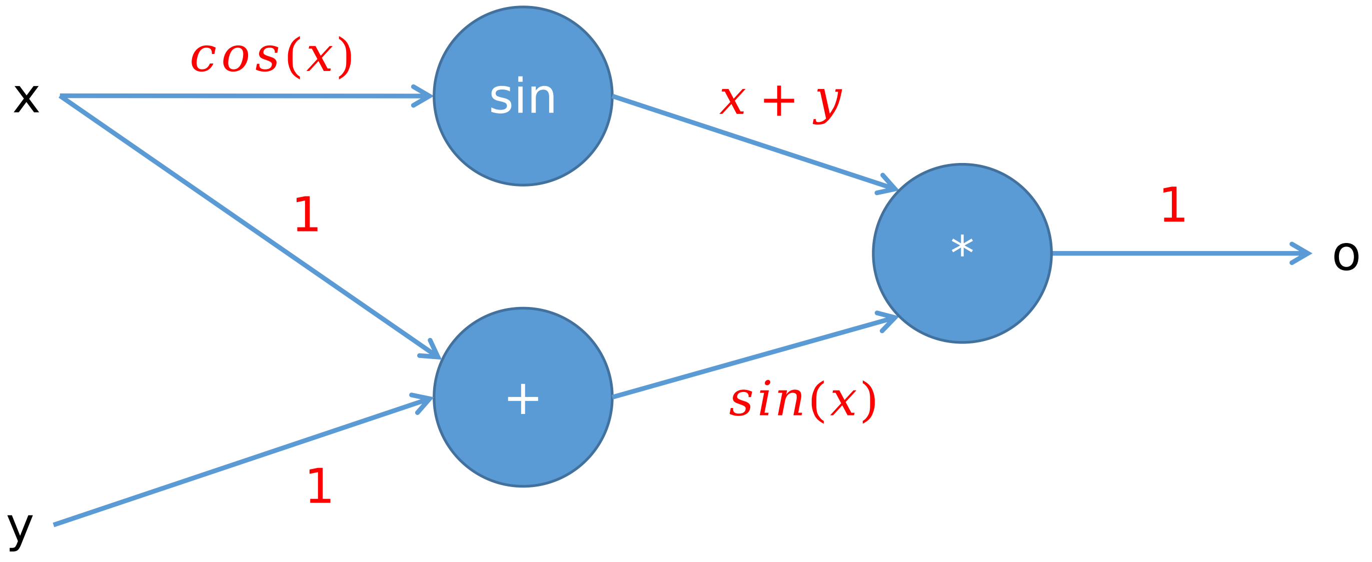


$$\frac{d(xy)}{dx} = y\frac{dx}{dx} + x\cancel{\frac{dy}{dx}} = y$$

$$\frac{d(xy)}{dy} = y\cancel{\frac{dx}{dy}} + x\frac{dy}{dy} = x$$

Computational Graphs: Example

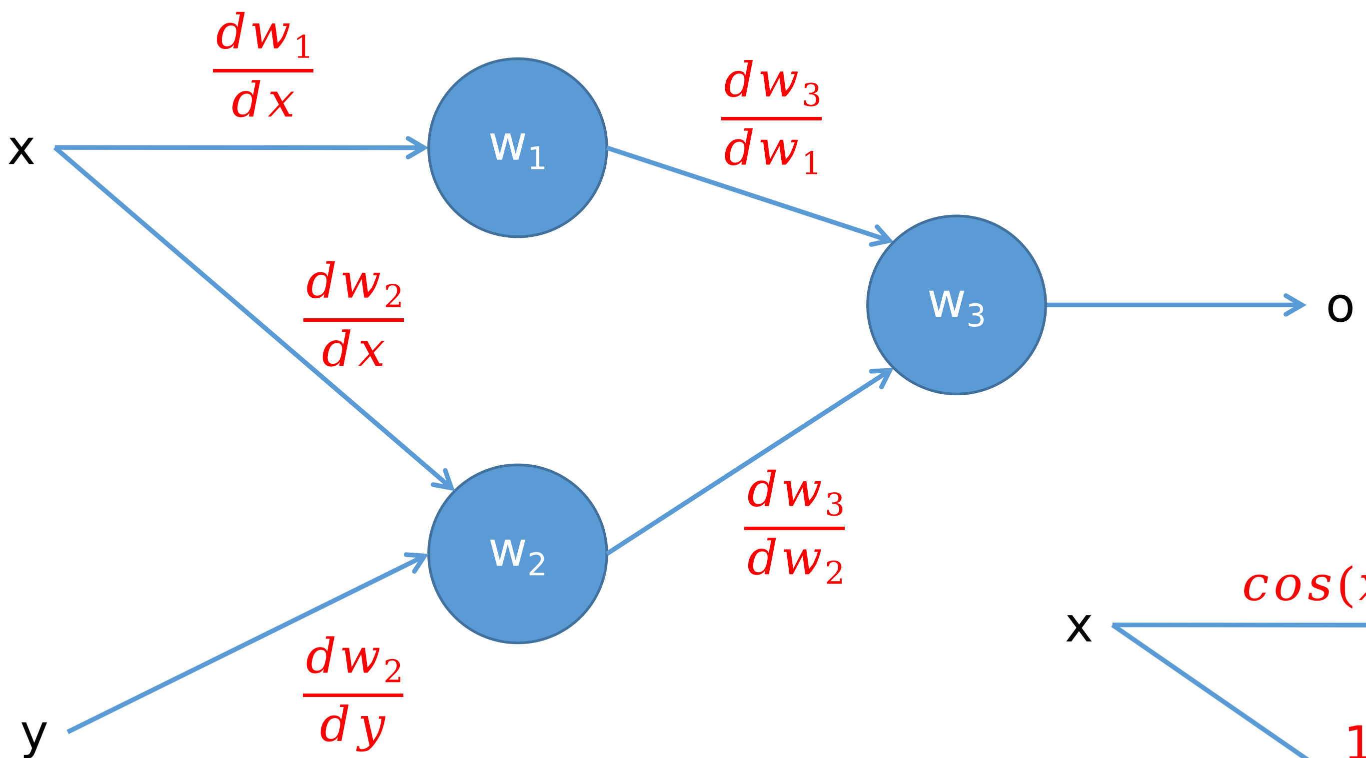
Let compute the Computational Graph of $o = (x + y)sin(x)$

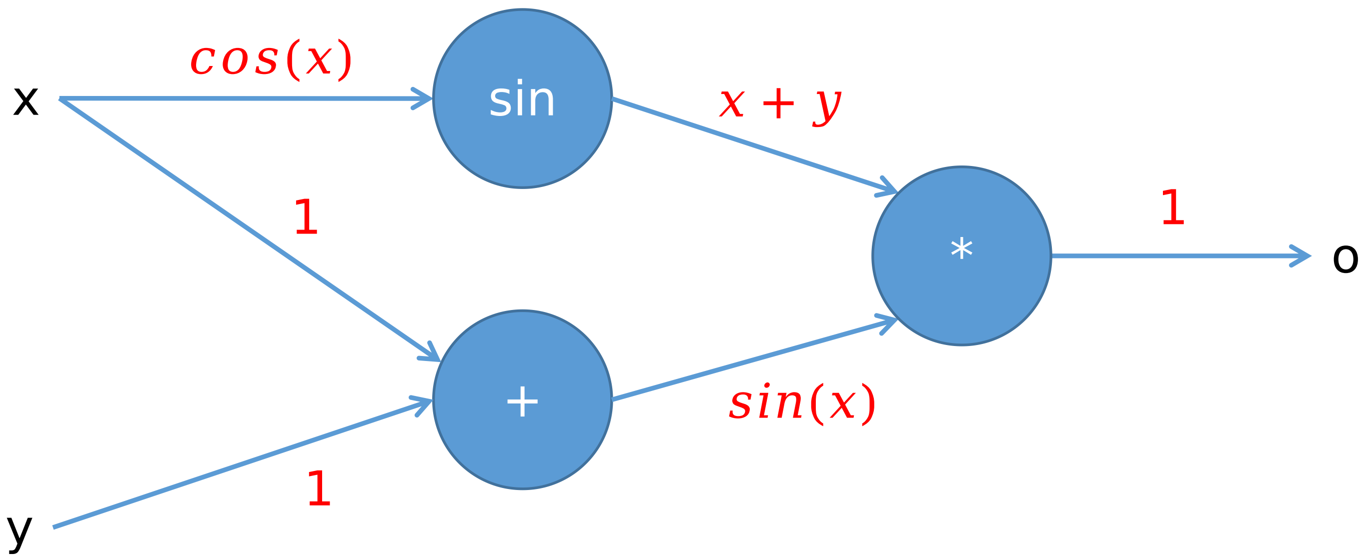


There are two ways of computing $\frac{do}{dx}$ and $\frac{do}{dy}$ using a computational graph:

- Forward Mode Differentiation
- Reverse Mode Differentiation

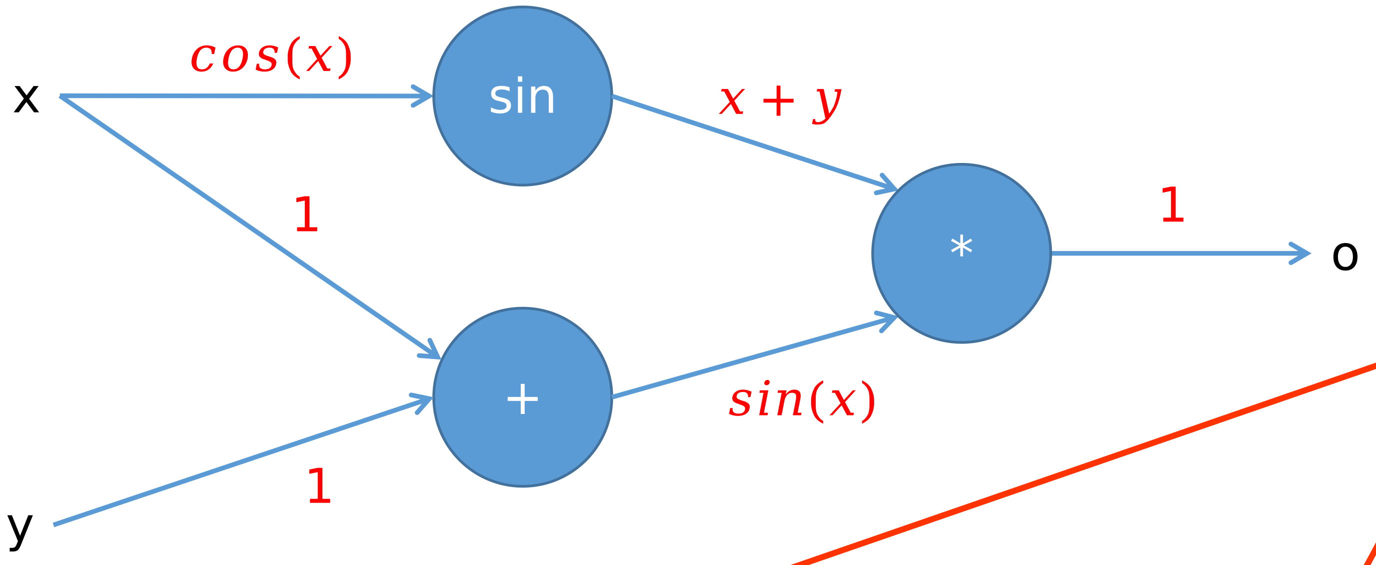
Computational Graphs: Generalization





Computational Graphs: Reverse Mode Differentiation

Lets start with the Reverse Mode Differentiation

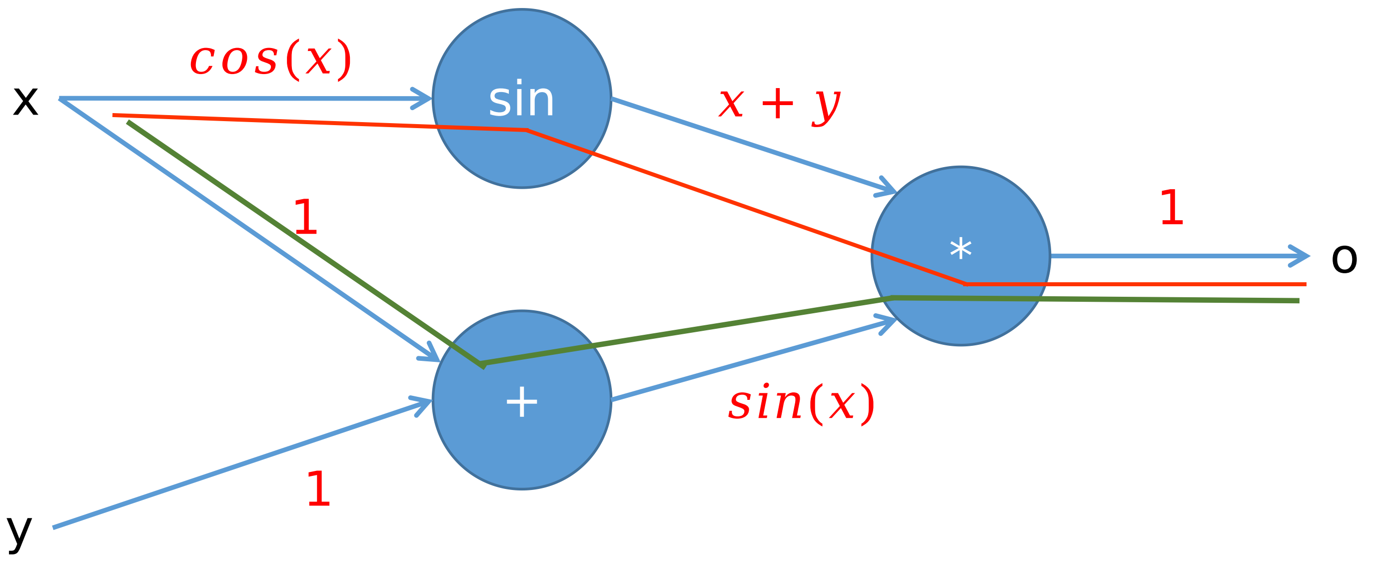


Because each path is a composition of operations

Because each split represent a binary operator with two derivation flows to be summed

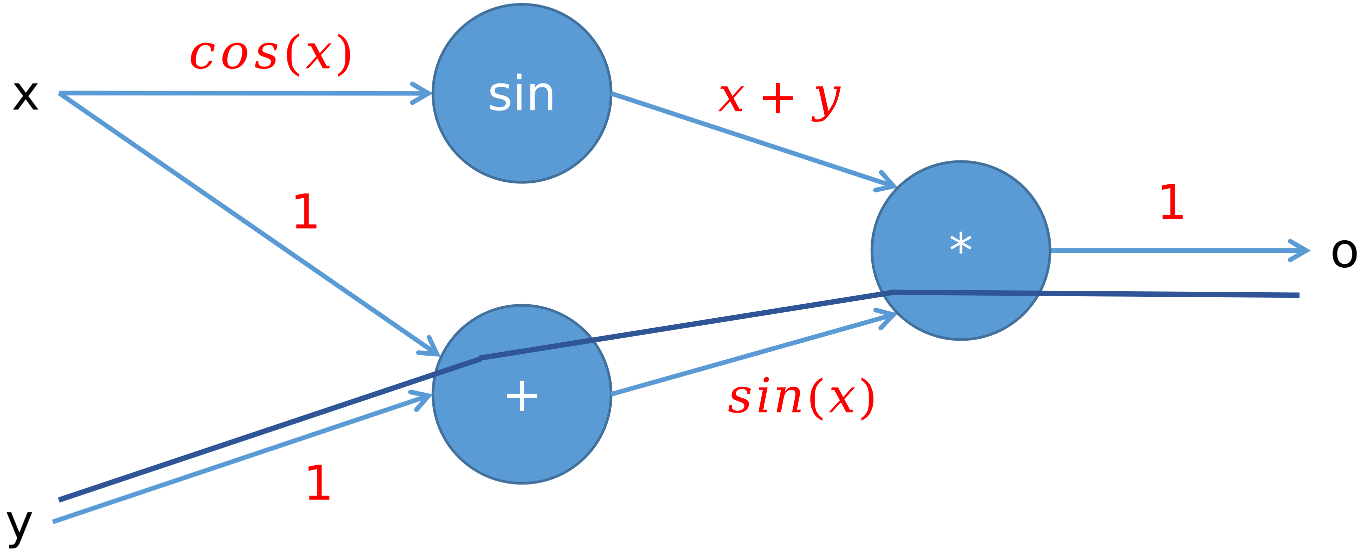
To compute the **derivative** of the **output** w.r.t. to one **input** you have to find **all possible paths** from the **output** to the **input** and **multiply the partial values over each path** and **sum all the results**

Computational Graphs: Example $\frac{do}{dx}$ in reverse mode



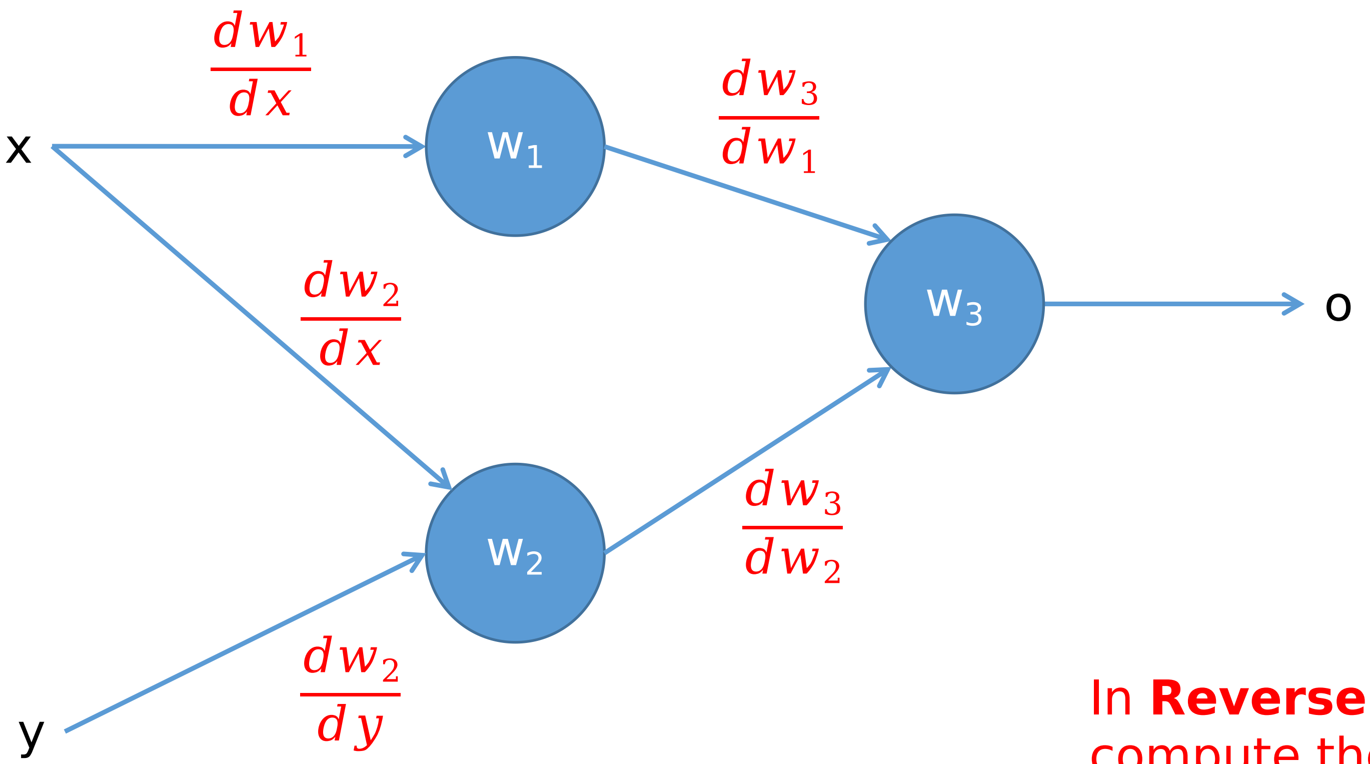
$$\frac{do}{dx} = (x + y)cos(x) + sin(x)$$

Computational Graphs: Example $\frac{do}{dy}$ in reverse mode



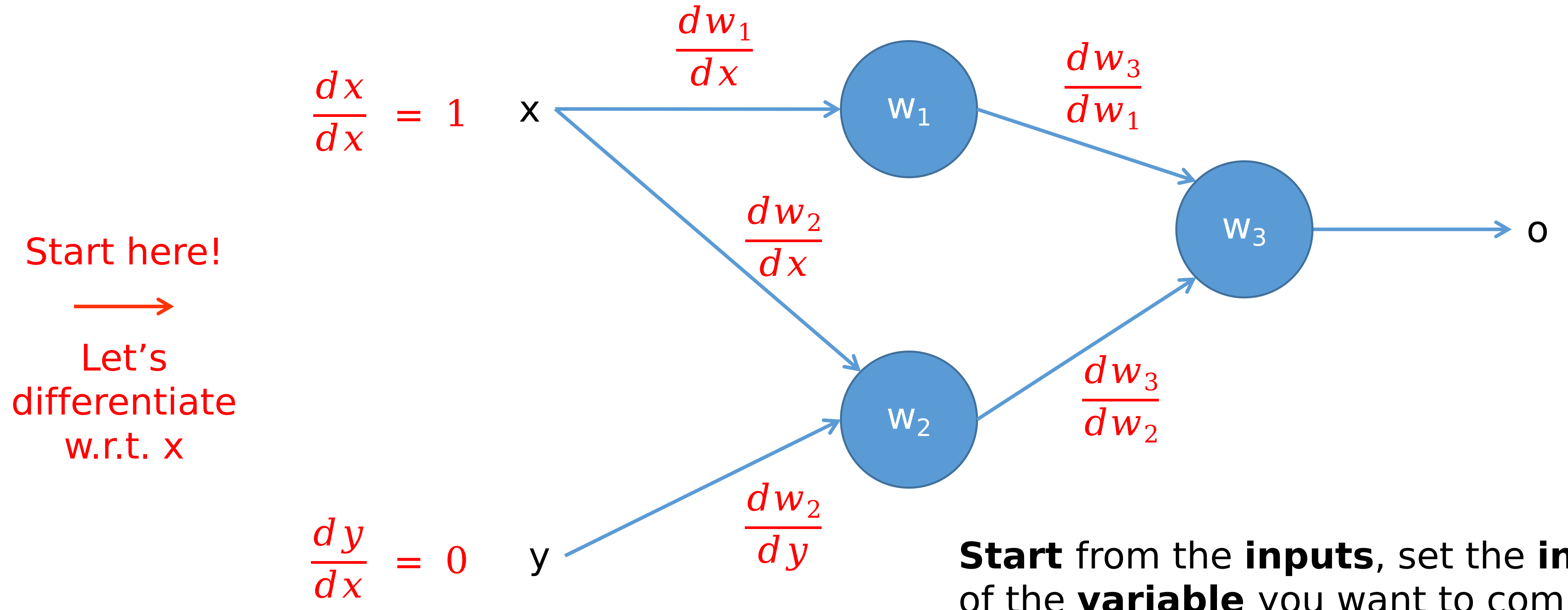
$$\frac{do}{dy} = sin(x)$$

Computational Graphs: Generalization Reverse Mode



In **Reverse Mode Differentiation** you can compute the **partial derivatives** of **one output** with respect to **every input** in **one pass**

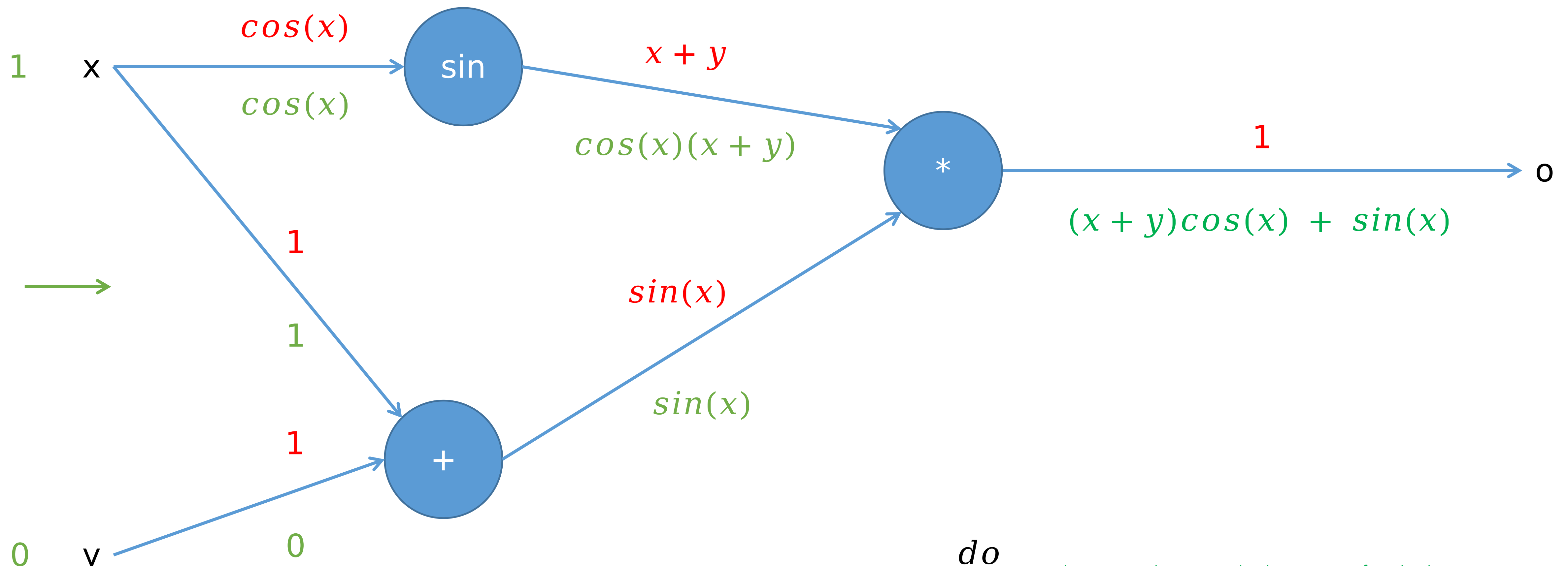
Computational Graphs: Forward Mode Differentiation



Start from the **inputs**, set the **initial derivative** of the **variable** you want to compute w.r.t. to **1**. Set the **other** derivatives to **0**.

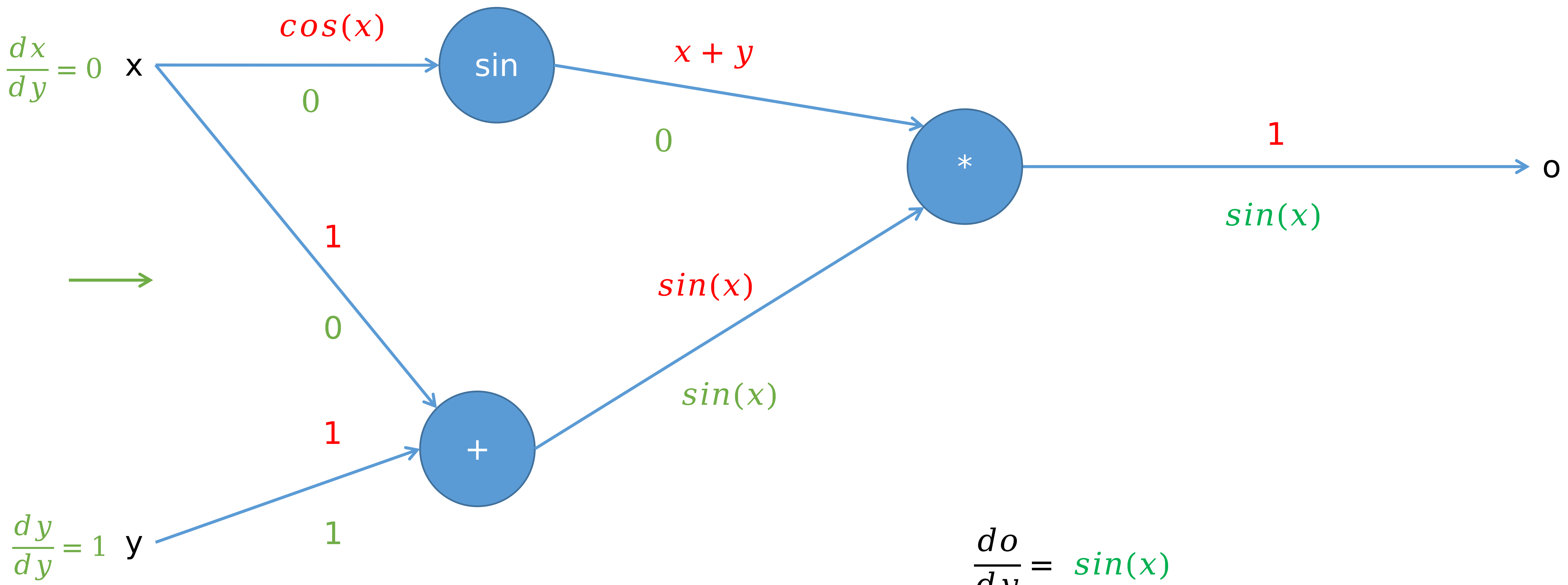
Compute forward **multiplying the values** over the **edges** and **summing** when **joining paths**

Computational Graphs: Example $\frac{do}{dx}$ in **forward mode**



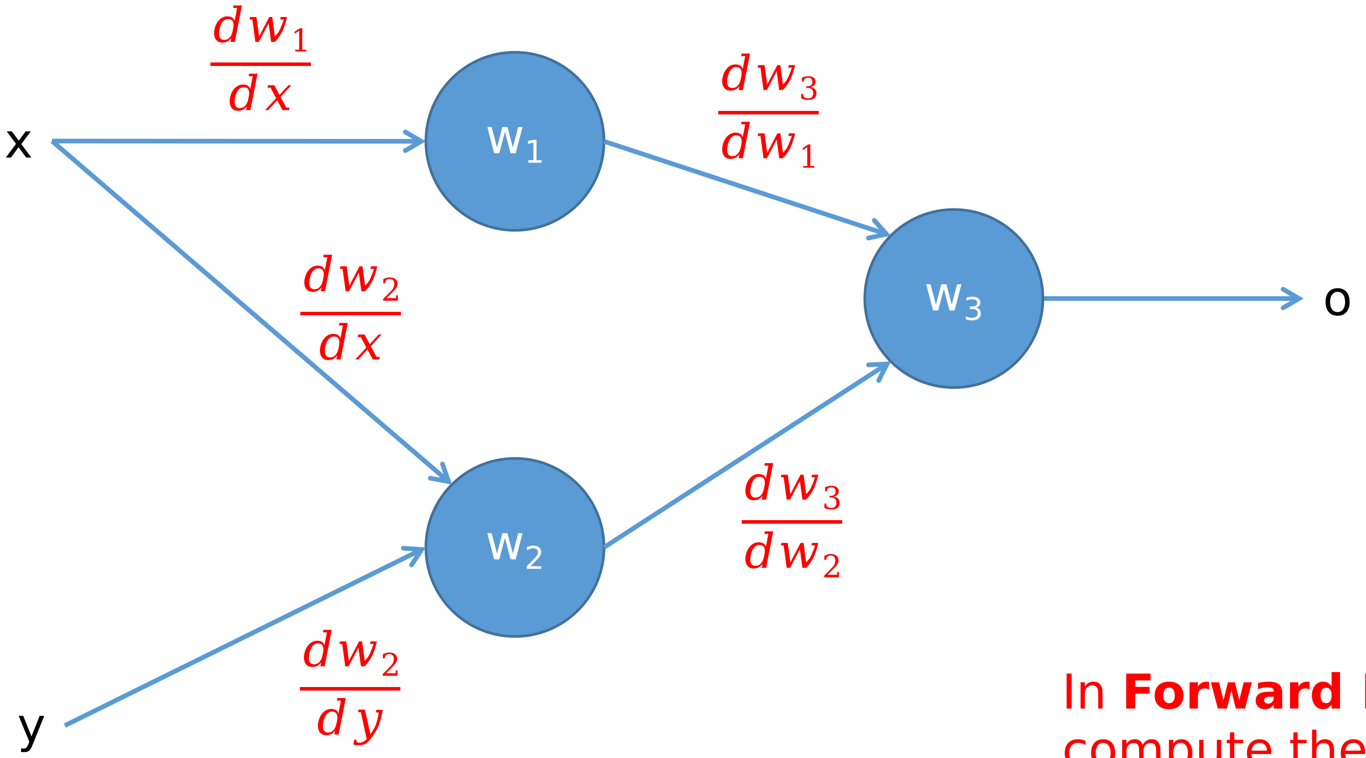
$$\frac{do}{dx} = (x + y)cos(x) + sin(x)$$

Computational Graphs: Example $\frac{do}{dy}$ in forward mode



$$\frac{do}{dy} = sin(x)$$

Computational Graphs: Generalization Forward Mode



In **Forward Mode Differentiation** you can compute the **partial derivatives** of **every output** with respect to **one input** in **one pass**

Computational Graphs: Trade-offs

Given a function

$$f\colon \mathbb{R}^n \to \mathbb{R}^m$$

<u>Reverse Mode Differentiation:</u>

- Each **edge** has **one derivative value**
- Derivative values in the CG are **shared** between partials
- You can compute the derivative of **one output** w.r.t. **every input** in **one pass**
- Ideal when $n \gg m$

<u>Forward Mode Differentiation:</u>

- Each **edge** has a **different derivative value** for each **partial**
- Derivative values in the CG are **not shared** between partials
- You can compute the derivative of **every output** w.r.t. **one input** in **one pass**
- Ideal when $n \ll m$

Machine learning error functions are: $f\colon \mathbb{R}^n \to \mathbb{R}$ (where n can be several billions)

Computational Graphs: Numerical Differentiation Frameworks

There are several **Numerical** Differentiation Frameworks

All these **Machine Learning frameworks** use reverse mode **computational graphs** under the hood.

They provide **nice** abstract **APIs** to **easily build complex** architectures.

Computational Graphs: PyTorch Example

Lets compute the derivative of $o = (x + y)sin(x)$ using PyTorch for $x = \pi$ and $y = \pi$

```
import torch
import math

x = torch.tensor([math.pi], requires_grad=True)
y = torch.tensor([math.pi], requires_grad=True)

o = (x+y)*torch.sin(x)
o.backward()

print(x.grad)
print(y.grad)
```

```
tensor([-6.2832])
tensor([-8.7423e-08])
```

$$\frac{do}{dx} = (x + y)cos(x) + sin(x)$$

$$\frac{do}{dx}(\pi, \pi) = (\pi + \pi)cos(\pi) + sin(\pi)$$

$$\frac{do}{dx}(\pi, \pi) = -2\pi$$

$$\frac{do}{dy} = sin(x)$$

$$\frac{do}{dy}(\pi, \pi) = sin(\pi) = 0$$

Exercises

Draw the computational graph and compute the derivative w.r.t. every input variable of the following functions using the reverse mode and the forward mode:

- $f(x)\ \mathbb{R} \rightarrow \mathbb{R} :\ ln(sin(x)\ \bullet\ cos(x))$

- $f(x,\ y,\ z,\ q)\ \mathbb{R}^4 \rightarrow \mathbb{R}^2 :\ \begin{pmatrix} (x+y)\bullet z \\ (x+y)\bullet q \end{pmatrix}$

- $f(x,\ y)\ \mathbb{R}^2 \rightarrow \mathbb{R} :\ (x+y)sin(x)\ \ where\ x = \pi\ and\ \ y = \pi \qquad (numerical)$