

[M]<sup>s</sup>

# 微分基本概念



ZOMI 酱

# 关于本课程

## 1. 课程背景

- AI框架中自动微分的重要性

## 2. 课程内容

- 微分基本概念：数值微分 - 符号微分 - 自动微分
- 自动微分模式：前向微分 - 后向微分 - 雅克比原理
- 具体实现方式：表达式或图 - 操作符重载OO - 源码转换 AST
- MindSpore实现：基于图表示的源码转换Graph Base AST
- 自动微分的未来
- 自动微分的挑战

# Overview

$$l_1 = x$$

$$l_{n+1} = 4l_n(1 - l_1)$$

$$f(x) = l_4 = 64x(1 - x)(1 - 2x)^2(1 - 8x + 8x^2)^2$$



手动计算微分 Manual Differentiation

$$f'(x)$$

$$\begin{aligned} &= 128x(1 - x)(-8 + 16x)(1 - 2x)^2(1 - 8x + 8x^2) + 64(1 - x)(1 - 2x)^2(1 - 8x + 8x^2)^2 \\ &\quad - 64x(1 - 2x)^2(1 - 8x + 8x^2)^2 - 256x(1 - x)(1 - 2x)(1 - 8x + 8x^2)^2 \end{aligned}$$

# Overview

$$l_1 = x$$
$$l_{n+1} = 4l_n(1 - l_n)$$
$$f(x) = l_4 = 64x(1 - x)(1 - 2x)^2(1 - 8x + 8x^2)^2$$

↓ 手动编写代码

```
f(x):  
  v = x  
  for i = 1 to 3:  
    v = 4 * v * (1 - v)  
  return v
```

```
f(x): return 64*x(1-x)*((1-2*x)^2)*(1-8*x+8*x*x)^2
```

自动微分

↓

```
f'(x):  
  (v, dv) = (x, 1)  
  for i = 1 to 3:  
    (v, dv) = (4 * v * (1 - v), 4*dv-8*v*dv)  
  return (v, dv)
```

手动计算微分

----->

符号微分

Closed-form

----->

数值微分

----->

$$f'(x)$$
$$= 128x(1 - x)(-8 + 16x)(1 - 2x)^2(1 - 8x + 8x^2)$$
$$+ 64(1 - x)(1 - 2x)^2(1 - 8x + 8x^2)^2 - 64x(1 - 2x)^2$$
$$(1 - 8x + 8x^2)^2 - 256x(1 - x)(1 - 2x)(1 - 8x + 8x^2)^2$$

↓

手动编写代码

```
f'(x):  
  return 128*x*(1-x)*(-8+16*x)*((1-2*x)^2)*(1-  
  8*x+8*x*x)+64*(1-x)*((1-2*x)^2)*((1-2*x+8*x*x)^2)-  
  (64*x*1-8*x)^2*(1-8*x+8*x*x)^2-256*x*(1-x)*(1-2*x)*(1-  
  8*x+8*x*x)^2
```

```
f'(x):  
  h = 0.000001  
  return (f(x + h) - f(x)) / h
```

# AD is not Symbolic Differentiation

符号微分：通过求导法则指定表达式变换规则

将原表达式转换为导师表达式：

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

$$\frac{d}{dx}(f(x) \cdot g(x)) \rightarrow \left(\frac{d}{dx}f(x)\right)g(x) + f(x)\left(\frac{d}{dx}g(x)\right)$$

**优势** • 精确数值结果      **缺点** • 表达式膨胀

$$f(x) = 64x(1-x)(1-2x)^2(1-8x+8x^2)^2$$

----->

$$\begin{aligned} f'(x) = & 128x(1-x)(-8+16x)(1-2x)^2(1-8x+8x^2) \\ & + 64(1-x)(1-2x)^2(1-8x+8x^2)^2 - 64x(1-2x)^2 \\ & (1-8x+8x^2)^2 - 256x(1-x)(1-2x)(1-8x+8x^2)^2 \end{aligned}$$

# AD is not Numerical Differentiation

## 数值微分：使用有限差分进行近似

对于多元函数  $f: \mathbb{R}^n \rightarrow \mathbb{R}$ ，其梯度为：

$$\nabla f = \left( \frac{\partial f}{\partial x_1}, \dots, \frac{\partial f}{\partial x_n} \right)$$

可以使用有限差分来近似：

$$\frac{\partial f(x)}{\partial x_i} \approx \frac{f(x + he_i) - f(x)}{h}, \text{ where } h > 0$$

- |           |        |           |  |
|-----------|--------|-----------|--|
| <b>优势</b> | • 容易实现 | <b>缺点</b> | <ul style="list-style-type: none"><li>• 计算结果不精确</li><li>• 计算复杂度高</li><li>• 对 <math>h</math> 的要求高</li></ul> |
|-----------|--------|-----------|--|

# AD is not Numerical Differentiation

## 数值微分：使用有限差分进行近似

对于多元函数  $f: \mathbb{R}^n \rightarrow \mathbb{R}$ ，其梯度为：

$$\nabla f = \left( \frac{\partial f}{\partial x_1}, \dots, \frac{\partial f}{\partial x_n} \right)$$

可以使用有限差分来近似：

$$\frac{\partial f(x)}{\partial x_i} \approx \frac{f(x + he_i) - f(x)}{h}, \text{ where } h > 0$$

**优势** • 容易实现

**缺点**

- 计算结果不精确  $\dashrightarrow$
- 计算复杂度高
- 对  $h$  的要求高
- 截断误差 ( Truncation Error )
- 舍入误差 ( Round-off Error )

# AD is not Numerical Differentiation

## 数值微分：使用有限差分进行近似

对于多元函数  $f: \mathbb{R}^n \rightarrow \mathbb{R}$ ，其梯度为：

$$\nabla f = \left( \frac{\partial f}{\partial x_1}, \dots, \frac{\partial f}{\partial x_n} \right)$$

可以使用有限差分来近似：

$$\frac{\partial f(x)}{\partial x_i} \approx \frac{f(x + he_i) - f(x)}{h}, \text{ where } h > 0$$

**优势** • 容易实现

**缺点**

- 计算结果不精确
  - 计算复杂度高
  - 对  $h$  的要求高
- >
- 阻碍其在ML领域发展



# What is AD



**自动微分：所有数值计算都由有限的基本运算组成**

**基本运算的导数表达式是已知的**

**通过链式法则将数值计算各部分组合成整体**

```
class Sigmoid(Primitive):

    @prim_attr_register
    def __init__(self, x):
        """Initialize Sigmoid."""

        output=1/(1+np.exp(-x))
        self.init_prim_io_names(inputs=['x'], outputs=['output'])
```

```
@bprop_getters.register(G.SigmoidGrad)
def get_bprop_sigmoid_grad(self):
    """Grad definition for `SigmoidGrad` operation."""
    sigmoid_grad = G.SigmoidGrad()

    def bprop(y, grad, out, dout):
        dy = dout * grad * (1. - 2 * y)
        dgrad = sigmoid_grad(y, dout)
        return dy, dgrad

    return bprop
```

# What is AD

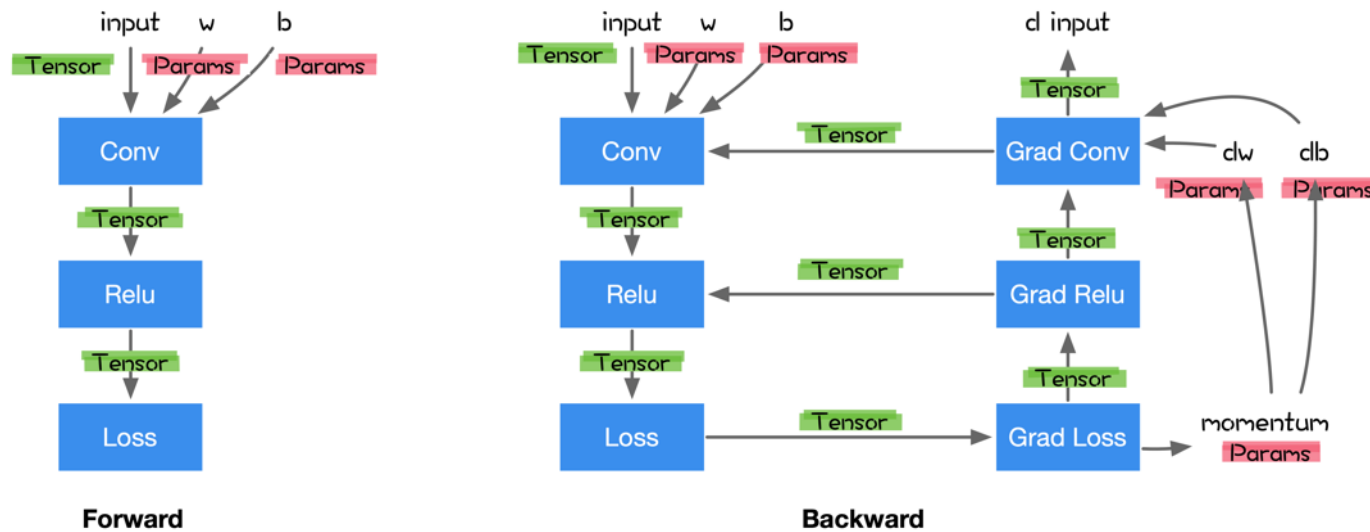
自动微分：所有数值计算都由有限的基本运算组成

基本运算的导数表达式是已知的

通过链式法则将数值计算各部分组合成整体

表达式追踪（Evaluation Trace）：追踪数值计算过程的中间变量

- 输入变量
- 中间变量
- 输出变量



# Overview

$$l_1 = x$$
$$l_{n+1} = 4l_n(1 - l_n)$$
$$f(x) = l_4 = 64x(1 - x)(1 - 2x)^2(1 - 8x + 8x^2)^2$$

↓ 手动编写代码

```
f(x):  
  v = x  
  for i = 1 to 3:  
    v = 4 * v * (1 - v)  
  return v
```

```
f(x): return 64*x(1-x)*((1-2*x)^2)*(1-8*x+8*x*x)^2
```

自动微分

↓

```
f'(x):  
  (v, dv) = (x, 1)  
  for i = 1 to 3:  
    (v, dv) = (4 * v * (1 - v), 4*dv-8*v*dv)  
  return (v, dv)
```

手动计算微分

----->

符号微分

Closed-form

----->

数值微分

----->

$$f'(x)$$
$$= 128x(1 - x)(-8 + 16x)(1 - 2x)^2(1 - 8x + 8x^2)$$
$$+ 64(1 - x)(1 - 2x)^2(1 - 8x + 8x^2)^2 - 64x(1 - 2x)^2$$
$$(1 - 8x + 8x^2)^2 - 256x(1 - x)(1 - 2x)(1 - 8x + 8x^2)^2$$

↓

手动编写代码

```
f'(x):  
  return 128*x*(1-x)*(-8+16*x)*((1-2*x)^2)*(1-  
  8*x+8*x*x)+64*(1-x)*((1-2*x)^2)*((1-2*x+8*x*x)^2)-  
  (64*x*1-8*x)^2*(1-8*x+8*x*x)^2-256*x*(1-x)*(1-2*x)*(1-  
  8*x+8*x*x)^2
```

```
f'(x):  
  h = 0.000001  
  return (f(x + h) - f(x)) / h
```

# Conclusion

1. 了解计算机微分的三种形式：符号微分、数值微分、自动微分；
2. 学习了符号微分的基本原理和优缺点；
3. 学习了数值微分的基本原理和优缺点；
4. 学习了自动微分的基本原理和与另外两种微分的差异；



BUILDING A BETTER CONNECTED WORLD

THANK YOU

EMTS

Copyright©2014 Huawei Technologies Co., Ltd. All Rights Reserved.

The information in this document may contain predictive statements including, without limitation, statements regarding the future financial and operating results, future product portfolio, new technology, etc. There are a number of factors that could cause actual results and developments to differ materially from those expressed or implied in the predictive statements. Therefore, such information is provided for reference purpose only and constitutes neither an offer nor an acceptance. Huawei may change the information at any time without notice.