Topics on Auto Differentiation

1. How it enables/speeds up ML frameworks
2. How it is related to differentiable programming
3. How both can be used for “simulation-based inference”

Machine Learning Seminar

Ze Ouyang

12th Nov, 2024

Overview of differentiation

- **Numerical differentiation:** $f'(x) \approx \dfrac{f(x+\delta x) - f(x)}{\delta x}$

  Pros: simple
  Cons: round-off & truncation errors, accuracy, inefficient for high-dimension

- **Symbolic differentiation:** Analytically derive $f'(x)$, like Mathematica

  Pros: exact
  Cons: computationally expensive, expression explosion

```
In[1]:= (*Define the function*) f[x_] := x^3 + 2 x^2 + x

        (*Compute the differential*)
        df = D[f[x], x]

        (*Display the result*)
        df

Out[2]= 1 + 4 x + 3 x^2

Out[3]= 1 + 4 x + 3 x^2
```

- **Auto differentiation:** Symbolic + Chain rule

  Pros: exact
  Cons: computation graph adding overhead

Preliminary Concepts on Auto Differentiation

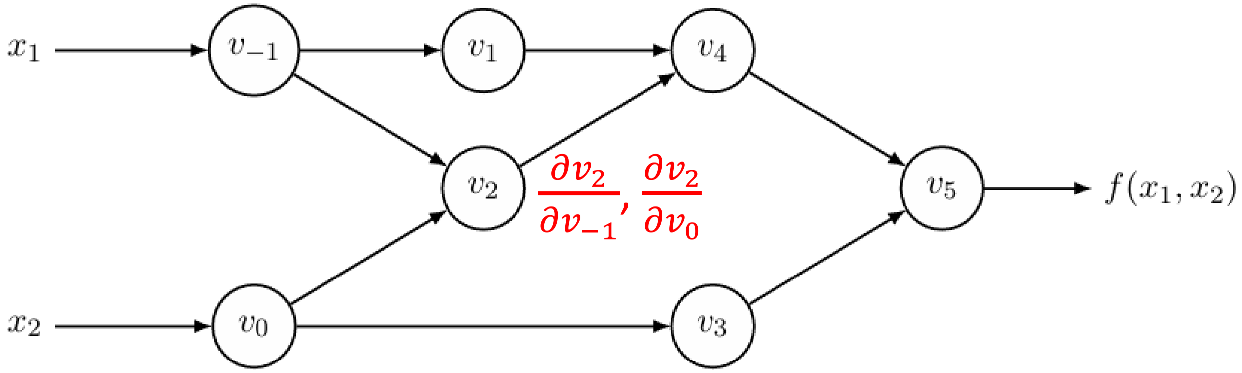
Input: $x_1, x_2, \ldots, x_n$
Output: $y_1, y_2, \ldots, y_m$ $\quad \mathbb{R}^n \Rightarrow \mathbb{R}^m$

$$J_f = \begin{bmatrix} \frac{\partial f_1}{\partial x_1} & \cdots & \frac{\partial f_1}{\partial x_n} \\ \vdots & \ddots & \vdots \\ \frac{\partial f_m}{\partial x_1} & \cdots & \frac{\partial f_m}{\partial x_n} \end{bmatrix}$$

Auto Differentiation
- Forward Mode: works better for $n < m$
- Backward Mode : works better for $n > m$

AD: Forward Mode

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



Computation graph

1. Divide into simple structures
2. Create computation graph (contains primal & derivatives respect to its last nodes)
3. Follow the direction and calculate gradient of a certain **input** variable

$v_4, \frac{\partial v_4}{\partial v_2}, \frac{\partial v_4}{\partial v_1}$, should have $\frac{\partial v_4}{\partial v_{-1}}$

$\frac{\partial v_4}{\partial v_{-1}} = \frac{\partial v_4}{\partial v_1}\frac{\partial v_1}{\partial v_{-1}}$ Chain rule

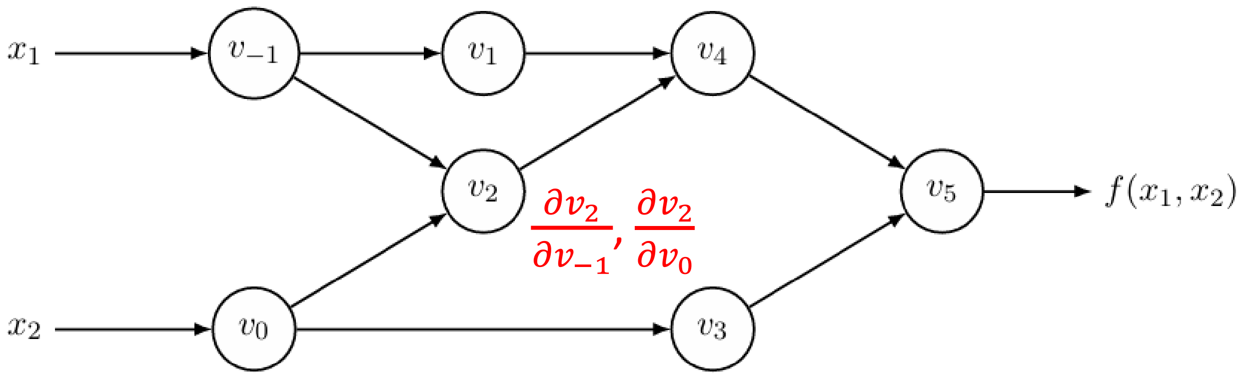
Forward Primal Trace		
v_{-1}	$= x_1$	$= 2$
v_0	$= x_2$	$= 5$
v_1	$= \ln v_{-1}$	$= \ln 2$
v_2	$= v_{-1} \times v_0$	$= 2 \times 5$
v_3	$= \sin v_0$	$= \sin 5$
v_4	$= v_1 + v_2$	$= 0.693 + 10$
v_5	$= v_4 - v_3$	$= 10.693 + 0.959$
y	$= v_5$	$= 11.652$

Forward Tangent (Derivative) Trace		
$\dot{v}_{-1}$	$= \dot{x}_1$	$= 1$
$\dot{v}_0$	$= \dot{x}_2$	$= 0$
$\dot{v}_1$	$= \dot{v}_{-1}/v_{-1}$	$= 1/2$
$\dot{v}_2$	$= \dot{v}_{-1} \times v_0 + \dot{v}_0 \times v_{-1}$	$= 1 \times 5 + 0 \times 2$
$\dot{v}_3$	$= \dot{v}_0 \times \cos v_0$	$= 0 \times \cos 5$
$\dot{v}_4$	$= \dot{v}_1 + \dot{v}_2$	$= 0.5 + 5$
$\dot{v}_5$	$= \dot{v}_4 - \dot{v}_3$	$= 5.5 - 0$
$\boldsymbol{\dot{y}}$	$\boldsymbol{= \dot{v}_5}$	$\boldsymbol{= 5.5}$

$$J_f = \begin{bmatrix} \frac{\partial f_1}{\partial x_1} & \cdots & \frac{\partial f_1}{\partial x_n} \\ \vdots & \ddots & \vdots \\ \frac{\partial f_m}{\partial x_1} & \cdots & \frac{\partial f_m}{\partial x_n} \end{bmatrix}$$

AD: Backward Mode

$$y = f(x_1, x_2) = \ln(x_1) + x_1x_2 + \sin(x_2)$$



Forward Primal Trace		
v_{-1}	$= x_1$	$= 2$
v_0	$= x_2$	$= 5$
v_1	$= \ln v_{-1}$	$= \ln 2$
v_2	$= v_{-1} \times v_0$	$= 2 \times 5$
v_3	$= \sin v_0$	$= \sin 5$
v_4	$= v_1 + v_2$	$= 0.693 + 10$
v_5	$= v_4 - v_3$	$= 10.693 + 0.959$
y	$= v_5$	$= 11.652$

Reverse Adjoint (Derivative) Trace			
$\boldsymbol{\bar{x}_1}$	$\boldsymbol{= \bar{v}_{-1}}$		$\mathbf{= 5.5}$
$\boldsymbol{\bar{x}_2}$	$\boldsymbol{= \bar{v}_0}$		$\mathbf{= 1.716}$
$\bar{v}_{-1}$	$= \bar{v}_{-1} + \bar{v}_1 \frac{\partial v_1}{\partial v_{-1}}$	$= \bar{v}_{-1} + \bar{v}_1 / v_{-1}$	$= 5.5$
$\bar{v}_0$	$= \bar{v}_0 + \bar{v}_2 \frac{\partial v_2}{\partial v_0}$	$= \bar{v}_0 + \bar{v}_2 \times v_{-1}$	$= 1.716$
$\bar{v}_{-1}$	$= \bar{v}_2 \frac{\partial v_2}{\partial v_{-1}}$	$= \bar{v}_2 \times v_0$	$= 5$
$\bar{v}_0$	$= \bar{v}_3 \frac{\partial v_3}{\partial v_0}$	$= \bar{v}_3 \times \cos v_0$	$= -0.284$
$\bar{v}_2$	$= \bar{v}_4 \frac{\partial v_4}{\partial v_2}$	$= \bar{v}_4 \times 1$	$= 1$
$\bar{v}_1$	$= \bar{v}_4 \frac{\partial v_4}{\partial v_1}$	$= \bar{v}_4 \times 1$	$= 1$
$\bar{v}_3$	$= \bar{v}_5 \frac{\partial v_5}{\partial v_3}$	$= \bar{v}_5 \times (-1)$	$= -1$
$\bar{v}_4$	$= \bar{v}_5 \frac{\partial v_5}{\partial v_4}$	$= \bar{v}_5 \times 1$	$= 1$
$\bar{v}_5$	$= \bar{y}$	$= 1$	

1. Divide into simple structures
2. Create computation graph (contains primal & derivatives respect to its last nodes)
3. Calculate from backward of a certain **output**

$v_5, \frac{\partial v_5}{\partial v_4}, \frac{\partial v_5}{\partial v_3}$, should have $\frac{\partial v_5}{\partial v_1}$

$\frac{\partial v_5}{\partial v_1} = \frac{\partial v_5}{\partial v_4}\frac{\partial v_4}{\partial v_1}$ Then go back to last v_{-1} node

Chain rule

$$J_f = \begin{bmatrix} \boxed{\frac{\partial f_1}{\partial x_1} \quad \cdots \quad \frac{\partial f_1}{\partial x_n}} \\ \vdots \quad \ddots \quad \vdots \\ \frac{\partial f_m}{\partial x_1} \quad \cdots \quad \frac{\partial f_m}{\partial x_n} \end{bmatrix}$$

Suitable for machine learning!

Differentiable Programming (DP)

$y = f(x; \theta)$

Make sure $f(x; \theta)$ is differentiable.

round(), floor(), ceil(), max(), min()… are not differentiable
for loop, if else,…are differentiable

DP enables complex programs to be differentiable by designing,

utilizing AD to make gradient-based optimization.

- Auto differentiation
- Continuous optimization
- Dynamic computation graph

Simulation-based Inference

- Statistical Inference

Given parameters θ, the observed data x is not deterministic, but we know probability

$\Downarrow$

Likelihood function

$\Downarrow$ maximize

Statistical analysis, find θ with confidence level

Example: radioactive matter half-time

$$N(t) = N_0 e^{-\lambda t}$$

- Simulation-based Inference (Likelihood-free)

Given parameters θ, the observed data x is deterministic, but hard to analytically relate

$\Downarrow$ Likelihood function is implicit

Loss function

$\Downarrow$ minimize

Find suitable θ

Example: From radiation pattern to probe electron beam inner structure

Conclusion

1. How AD enables/speeds up ML frameworks

    - By minimizing **loss function** with info of parameter gradient.

2. How AD is related to differentiable programming

    - AD is a core part of DP

3. How both can be used for “simulation-based inference”

$$y = f(x;\theta)$$

- How to minimize loss function? Genetic algorithm, differential evolution, & Gradient descent
- Fit $\theta = f^{-1}(x;y)$ with Neural Network, which is a black box. AD is extensively used in training the neural network.