System for Artificial Intelligence

Automatic Differentiation

Siyuan Feng
Shanghai Innovation Institute

OUTLINE

01 Differentiation Methods

02 Reverse Mode Auto Diff

01

Differentiation Methods

Numerical Differentiation

上海创智学院
Shanghai Innovation Institute

Directly compute the partial gradient by definition

$$\frac{\partial f(\theta)}{\partial \theta_i} = \lim_{\epsilon \to 0} \frac{f(\theta + \epsilon e_i) - f(\theta)}{\epsilon}$$

A more numerically accurate way to approximate the gradient

$$\frac{\partial f(\theta)}{\partial \theta_i} = \frac{f(\theta + \epsilon e_i) - f(\theta - \epsilon e_i)}{2\epsilon} + o(\epsilon^2)$$

Suffer from **numerical error**, **less efficient to compute**

Numerical Gradient Checking

However, numerical differentiation is a powerful tool to verify an implement of an automatic differentiation algorithm

$$\delta^T \nabla_\theta f(\theta) = \frac{f(\theta + \epsilon\delta) - f(\theta - \epsilon\delta)}{2\epsilon} + o(\epsilon^2)$$

Pick δ from unit ball, check the above invariance.

Symbolic Differentiation

上海创智学院
Shanghai Innovation Institute

Use the model formula to derive gradients by sum, product and chain rules

- $\frac{\partial\left(f(\theta)+g(\theta)\right)}{\partial\theta} =$
- $\frac{\partial\left(f(\theta)\cdot g(\theta)\right)}{\partial\theta} =$
- $\frac{\partial\left(f\left(g(\theta)\right)\right)}{\partial\theta} =$

Symbolic Differentiation

上海创智学院
Shanghai Innovation Institute

Use the model formula to derive gradients by sum, product and chain rules

- $\frac{\partial(f(\theta)+g(\theta))}{\partial\theta} = \frac{\partial(f(\theta))}{\partial\theta} + \frac{\partial(g(\theta))}{\partial\theta}$
- $\frac{\partial(f(\theta)\cdot g(\theta))}{\partial\theta} = g(\theta)\frac{\partial(f(\theta))}{\partial\theta} + f(\theta)\frac{\partial(g(\theta))}{\partial\theta}$
- $\frac{\partial(f(g(\theta)))}{\partial\theta} = \frac{\partial(f(g(\theta)))}{\partial g(\theta)}\frac{\partial(g(\theta))}{\partial\theta}$

Symbolic Differentiation

上海创智学院
Shanghai Innovation Institute

Naively method can resultin wasted computation

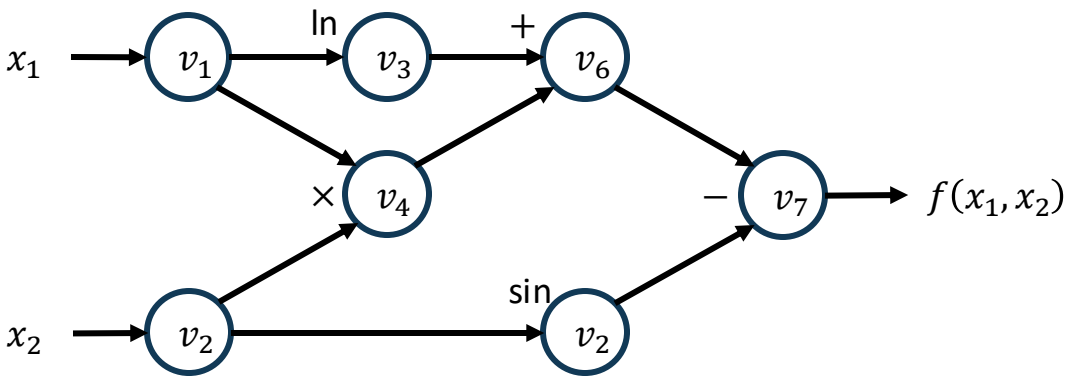
Example:

$$f(\theta) = \prod_{i=1}^{n} \theta_i, \qquad \frac{f(\theta)}{\partial \theta_k} = \prod_{i \neq k}^{n} \theta_i$$

Cost $n(n-2)$ multiplies to compute all partial gradients

Recap: Computational Graph

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

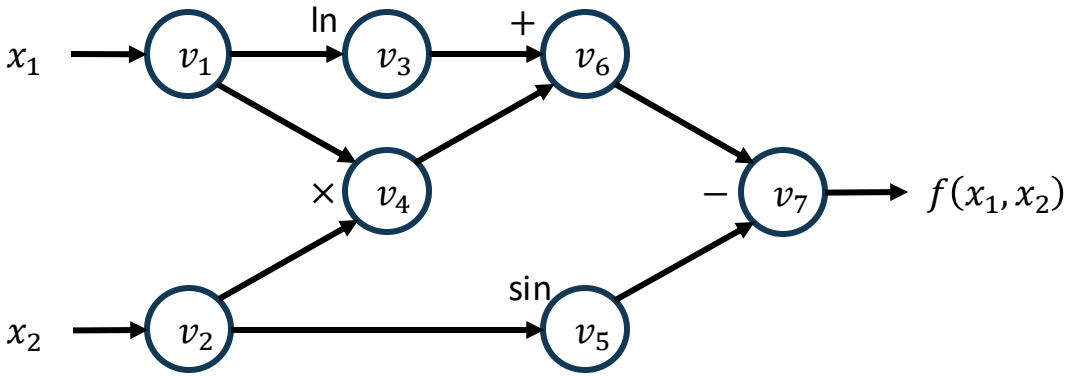


Each node represent an (intermediate) value in the computation. Edges present input output relations.

Example based on A. G. Baydin, B. A. Pearlmutter, A. A. Radul, and J. M. Siskind. Automatic differentiation in machine learning: a survey. JMLR 2018

Forward Mode Automatic Differentiation

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



$$v_1 = x_1 = 2$$
$$v_2 = x_2 = 5$$
$$v_3 = \ln(v_1) = 0.693$$
$$v_4 = v_1 \times v_2 = 10$$
$$v_5 = \sin(v_2) = -0.959$$
$$v_6 = v_3 + v_4 = 10.693$$
$$v_7 = v_6 - v_5 = 11.652$$

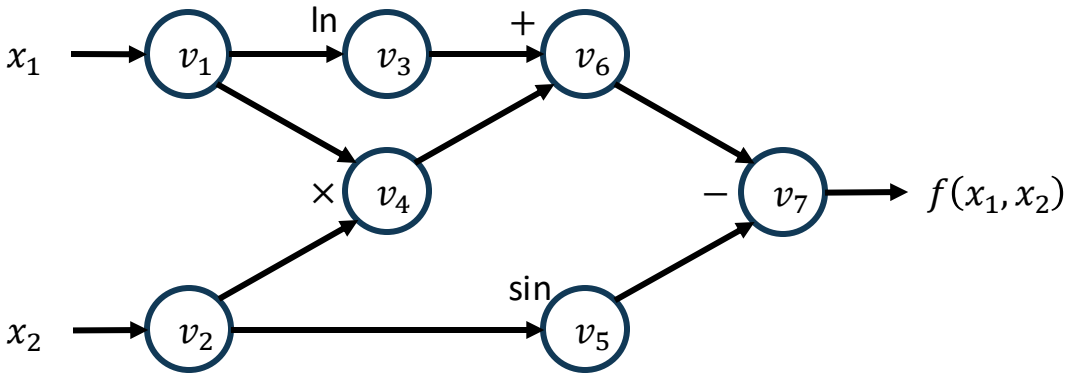
Define $\dot{v}_i = \frac{\partial v_i}{\partial x_1}$

We can then compute the $\dot{v}_i$ iteratively in the forward topological order of the computational graph

$$\dot{v}_1 = 1$$
$$\dot{v}_2 = 0$$
$$\dot{v}_3 = \frac{\dot{v}_1}{v_1} = 0.5$$
$$\dot{v}_4 = \dot{v}_1 v_2 + \dot{v}_2 v_1 = 5$$
$$\dot{v}_5 = \dot{v}_2 \cos(v_2) = 0$$
$$\dot{v}_6 = \dot{v}_3 + \dot{v}_4 = 5.5$$
$$\dot{v}_7 = \dot{v}_6 - \dot{v}_5 = 5.5$$
$$\frac{\partial f(x_1, x_2)}{\partial x_1} = \dot{v}_7 = 5.5$$

Forward Mode Automatic Differentiation

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



$$v_1 = x_1 = 2$$
$$v_2 = x_2 = 5$$
$$v_3 = \ln(v_1) = 0.693$$
$$v_4 = v_1 \times v_2 = 10$$
$$v_5 = \sin(v_2) = -0.959$$
$$v_6 = v_3 + v_4 = 10.693$$
$$v_7 = v_6 - v_5 = 11.652$$

Define $\dot{v}_i = \frac{\partial v_i}{\partial x_2}$

$$\dot{v}_1 = 0$$
$$\dot{v}_2 = 1$$
$$\dot{v}_3 = \frac{\dot{v}_1}{v_1} = 0$$
$$\dot{v}_4 = \dot{v}_1 v_2 + \dot{v}_2 v_1 = 2$$
$$\dot{v}_5 = \dot{v}_2 \cos(v_2) = 0.284$$
$$\dot{v}_6 = \dot{v}_3 + \dot{v}_4 = 2$$
$$\dot{v}_7 = \dot{v}_6 - \dot{v}_5 = 1.716$$

$$\frac{\partial f(x_1, x_2)}{\partial x_2} = \dot{v}_7 = 1.716$$

Limitations of Forward Mode

上海创智学院
Shanghai Innovation Institute

- For $f: \mathbb{R}^n \to \mathbb{R}^k$, we need n forward propagation passes to get the gradient with respect to each input.

- However, we mostly care where $k = 1$ and a large n.

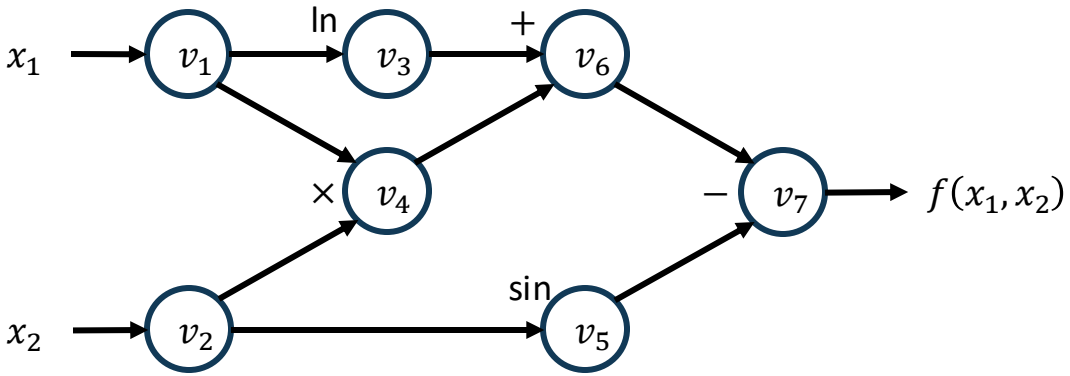
- The time complexity of forward mode auto diff is $O(N^2)$

02

Reverse Mode Auto Diff

Reverse Mode Automatic Differentiation

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



$$v_1 = x_1 = 2$$
$$v_2 = x_2 = 5$$
$$v_3 = \ln(v_1) = 0.693$$
$$v_4 = v_1 \times v_2 = 10$$
$$v_5 = \sin(v_2) = -0.959$$
$$v_6 = v_3 + v_4 = 10.693$$
$$v_7 = v_6 - v_5 = 11.652$$

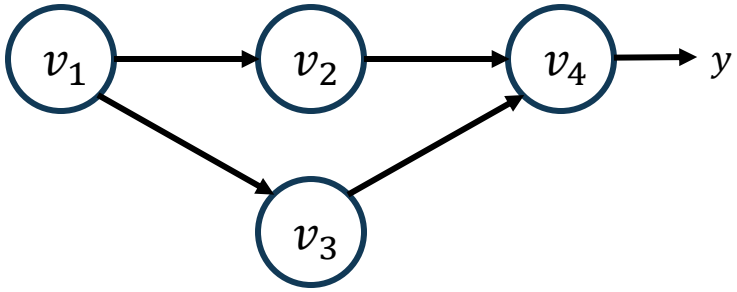
Get all input gradients in one pass

Define $\bar{v}_i = \frac{\partial f(x_1, x_2)}{\partial v_i}$, while forward mode is $\dot{v}_i = \frac{\partial v_i}{\partial x_1}$

We can then compute the $\bar{v}_i$ iteratively in the **reverse** topological order of the computational graph

$$\bar{v}_7 = \frac{\partial f(x_1, x_2)}{\partial v_7} = 1$$
$$\bar{v}_6 = \bar{v}_7 \cdot \frac{\partial v_7}{\partial v_6} = \bar{v}_7 \cdot 1 = 1$$
$$\bar{v}_5 = \bar{v}_7 \cdot \frac{\partial v_7}{\partial v_5} = \bar{v}_7 \cdot -1 = -1$$
$$\bar{v}_4 = \bar{v}_6 \cdot \frac{\partial v_6}{\partial v_4} = \bar{v}_6 \cdot 1 = 1$$
$$\bar{v}_3 = \bar{v}_6 \cdot \frac{\partial v_6}{\partial v_3} = \bar{v}_6 \cdot 1 = 1$$
$$\bar{v}_2 = \bar{v}_5 \cdot \frac{\partial v_5}{\partial v_2} + \bar{v}_4 \cdot \frac{\partial v_4}{\partial v_2} = \bar{v}_5 \cdot \cos(v_2) + \bar{v}_4 \cdot v_1 = 1.716$$
$$\bar{v}_1 = \bar{v}_4 \cdot \frac{\partial v_4}{\partial v_1} + \bar{v}_3 \cdot \frac{\partial v_3}{\partial v_1} = \bar{v}_4 \cdot v_2 + \bar{v}_2 \cdot \frac{1}{v_1} = 5.5$$

Derivation for Branches



In this case, v_1 is being used in multiple pathways (i.e, v_2 and v_3)

y can be written in form of $y = f(v_2, v_3)$, hence

$$\bar{v}_1 = \frac{\partial y}{\partial v_1} = \frac{\partial f(v_2, v_3)}{\partial v_1} = \frac{\partial f(v_2, v_3)}{\partial v_2} \cdot \frac{\partial v_2}{\partial v_1} + \frac{\partial f(v_2, v_3)}{\partial v_3} \cdot \frac{\partial v_3}{\partial v_1} = \bar{v}_2 \cdot \frac{\partial v_2}{\partial v_1} + \bar{v}_3 \cdot \frac{\partial v_3}{\partial v_1}$$

Define partial adjoint $\bar{v}_{i \to j} = \bar{v}_j \cdot \frac{\partial v_j}{\partial v_i}$ for each input output node pair i and j, we have

$$\bar{v}_i = \sum_{j \in adjoints(i)} \bar{v}_{i \to j}$$

We can compute partial adjoints separately then sum them together

Reverse Automatic Differentiation Algorithm

上海创智学院 Shanghai Innovation Institute

```
def gradient(out):
    node_to_grad = {out: [1]}
```

node_to_grad: dictionary to record a list of partial adjoints for each node

```
    for i in reverse_topo_order(out):
        v̄_i = Σ_j v̄_{i→j} =sum(node_to_grad[i])
```

Sum up partial adjoints

```
        for k in inputs[i]
            compute v̄_{k→i} = v̄_i ∂v_i/∂v_k
            append v̄_{k→i} to node_to_grad[k]
    return adjoint of input v̄_input
```

Propagates partial adjoint to its input node

Reverse Mode AD by Extending Computational Graph

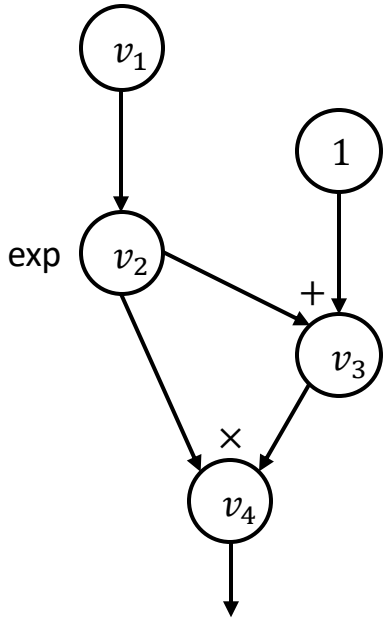
上海创智学院 Shanghai Innovation Institute

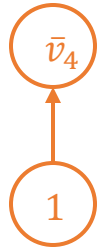
```
def gradient(out):
    node_to_grad = {out: [1]}
    for i in reverse_topo_order(out):
        v̄_i = Σ_j v̄_{i→j} =sum(node_to_grad[i])
        for k in inputs[i]
            compute v̄_{k→i} = v̄_i ∂v_i/∂v_k
            append v̄_{k→i} to node_to_grad[k]
    return adjoint of input v̄_input
```

$i = 4$

node_to_grad:{
    4: [1]
}

computed $\bar{v}_i$:{
    $\bar{v}_4$
}





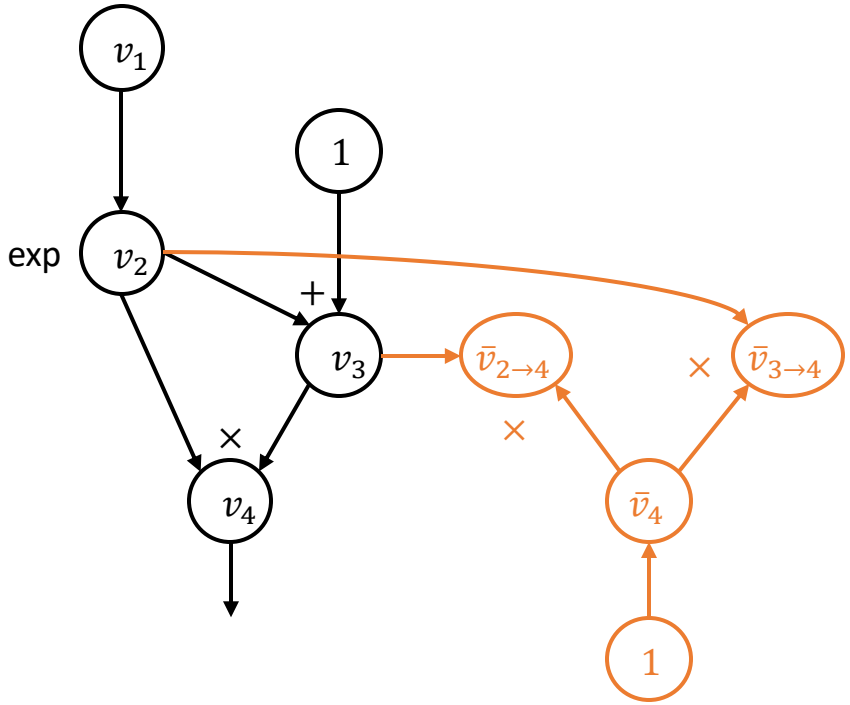
Reverse Mode AD by Extending Computational Graph


```
def gradient(out):
    node_to_grad = {out: [1]}
    for i in reverse_topo_order(out):
        v̄_i = Σ_j v̄_{i→j} =sum(node_to_grad[i])
        for k in inputs[i]
            compute v̄_{k→i} = v̄_i ∂v_i/∂v_k
            append v̄_{k→i} to node_to_grad[k]
    return adjoint of input v̄_input
```

$i = 4$

node_to_grad:{
  2: $[\bar{v}_{2\to4}]$
  3: $[\bar{v}_{3\to4}]$
  4: [1]
}

computed $\bar{v}_i$:{
  $\bar{v}_4$
}



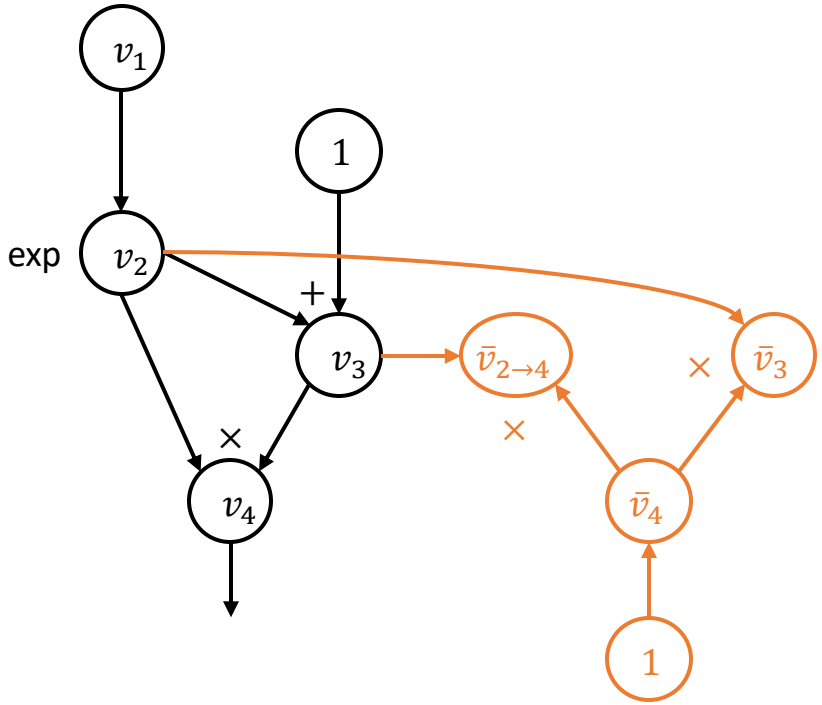
Reverse Mode AD by Extending Computational Graph

```
def gradient(out):
    node_to_grad = {out: [1]}
    for i in reverse_topo_order(out):
        v̄_i = Σ_j v̄_{i→j} =sum(node_to_grad[i])
        for k in inputs[i]
            compute v̄_{k→i} = v̄_i ∂v_i/∂v_k
            append v̄_{k→i} to node_to_grad[k]
    return adjoint of input v̄_input
```

$i = 3$

node_to_grad:{
  2: $[\bar{v}_{2\to4}]$
  3: $[\bar{v}_{3\to4}]$
  4: [1]
}

computed $\bar{v}_i$:{
  $\bar{v}_3$
  $\bar{v}_4$
}



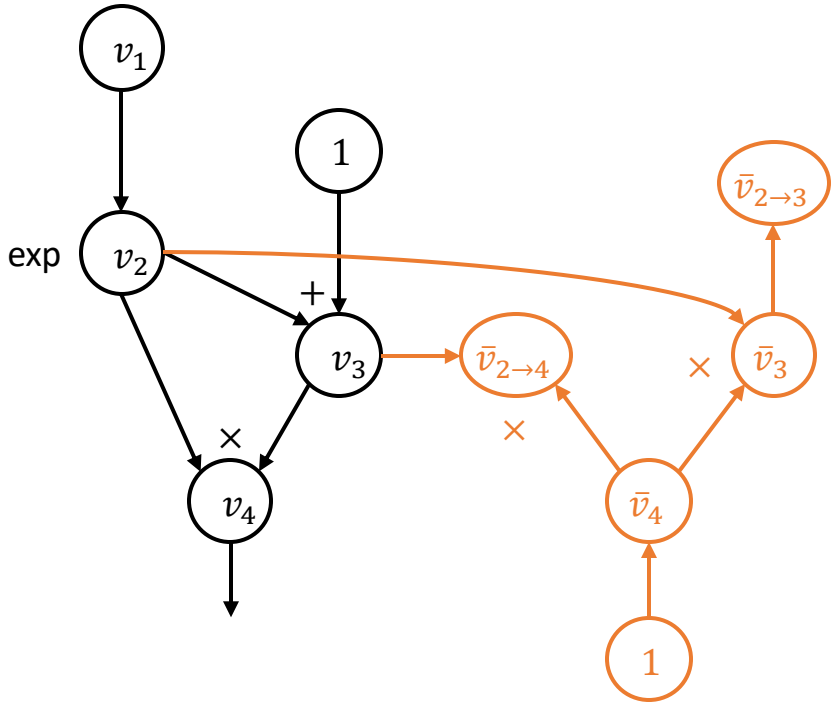
Reverse Mode AD by Extending Computational Graph


```
def gradient(out):
    node_to_grad = {out: [1]}
    for i in reverse_topo_order(out):
        v̄_i = Σ_j v̄_{i→j} =sum(node_to_grad[i])
        for k in inputs[i]
            compute v̄_{k→i} = v̄_i ∂v_i/∂v_k
            append v̄_{k→i} to node_to_grad[k]
    return adjoint of input v̄_input
```

$i = 3$

node_to_grad:{
  2: $[\bar{v}_{2\to4}, \bar{v}_{2\to3}]$
  3: $[\bar{v}_{3\to4}]$
  4: [1]
}

computed $\bar{v}_i$:{
  $\bar{v}_3$
  $\bar{v}_4$
}



Reverse Mode AD by Extending Computational Graph

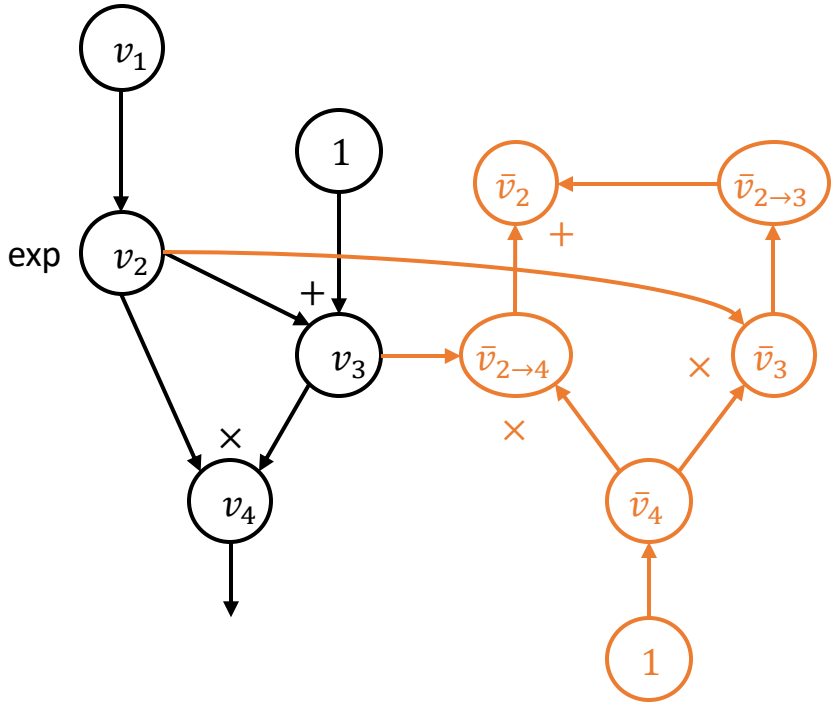
上海创智学院 Shanghai Innovation Institute

```
def gradient(out):
    node_to_grad = {out: [1]}
    for i in reverse_topo_order(out):
        v̄_i = Σ_j v̄_{i→j} =sum(node_to_grad[i])
        for k in inputs[i]
            compute v̄_{k→i} = v̄_i ∂v_i/∂v_k
            append v̄_{k→i} to node_to_grad[k]
    return adjoint of input v̄_input
```

$i = 2$

node_to_grad:{
  2: $[\bar{v}_{2\to4}, \bar{v}_{2\to3}]$
  3: $[\bar{v}_{3\to4}]$
  4: [1]
}

computed $\bar{v}_i$:{
  $\bar{v}_2$
  $\bar{v}_3$
  $\bar{v}_4$
}



Reverse Mode AD by Extending Computational Graph

上海创智学院 Shanghai Innovation Institute

```
def gradient(out):
    node_to_grad = {out: [1]}
    for i in reverse_topo_order(out):
        v̄_i = Σ_j v̄_{i→j} =sum(node_to_grad[i])
        for k in inputs[i]
            compute v̄_{k→i} = v̄_i ∂v_i/∂v_k
            append v̄_{k→i} to node_to_grad[k]
    return adjoint of input v̄_input
```

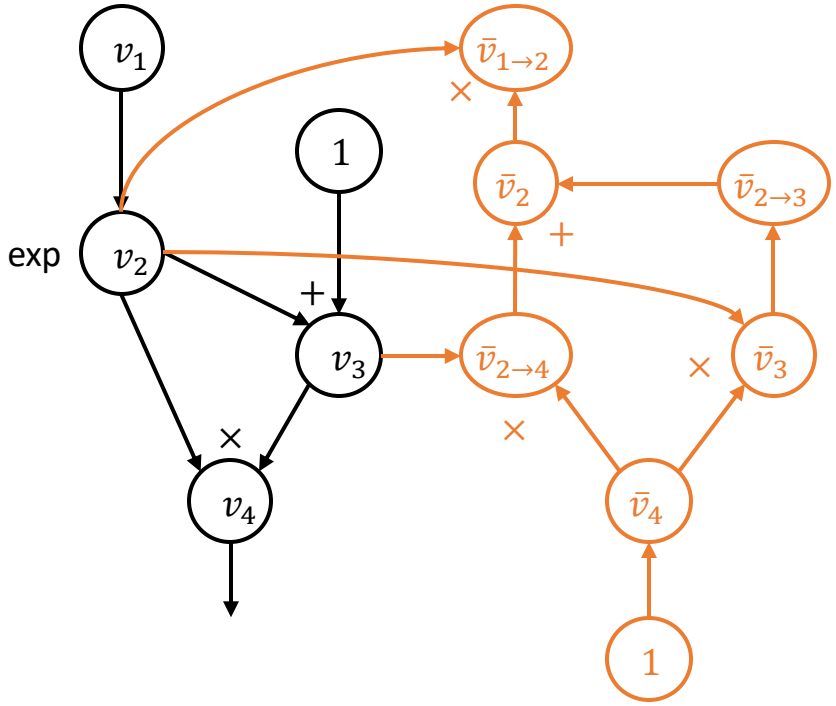
$i = 2$

node_to_grad:{
- 1: $[\bar{v}_{1\to 2}]$
- 2: $[\bar{v}_{2\to 4}, \bar{v}_{2\to 3}]$
- 3: $[\bar{v}_{3\to 4}]$
- 4: $[1]$

}

computed $\bar{v}_i$:{
- $\bar{v}_2$
- $\bar{v}_3$
- $\bar{v}_4$

}



Reverse Mode AD by Extending Computational Graph

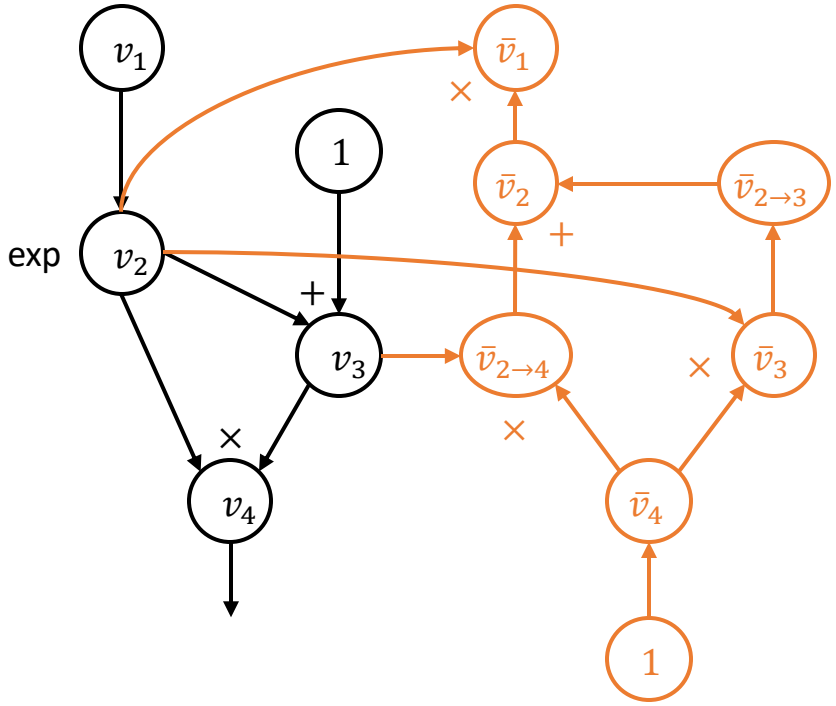
```
def gradient(out):
    node_to_grad = {out: [1]}
    for i in reverse_topo_order(out):
        v̄_i = Σ_j v̄_{i→j} =sum(node_to_grad[i])
        for k in inputs[i]
            compute v̄_{k→i} = v̄_i ∂v_i/∂v_k
            append v̄_{k→i} to node_to_grad[k]
    return adjoint of input v̄_input
```

$i = 1$

node_to_grad:{
- 1: $[\bar{v}_{1\to 2}]$
- 2: $[\bar{v}_{2\to 4}, \bar{v}_{2\to 3}]$
- 3: $[\bar{v}_{3\to 4}]$
- 4: $[1]$

}

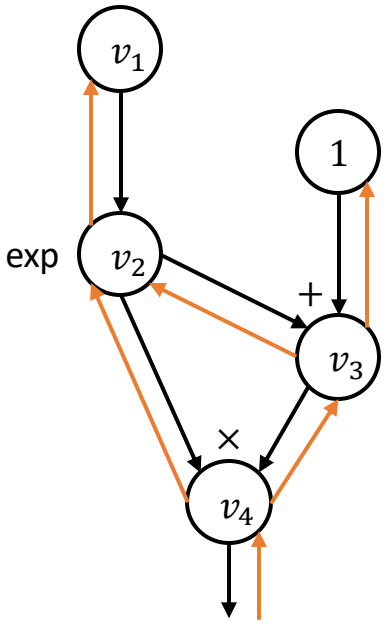
computed $\bar{v}_i$:{
$\bar{v}_1$
$\bar{v}_2$
$\bar{v}_3$
$\bar{v}_4$
}



Compute in-place vs. Extend Computational Graph

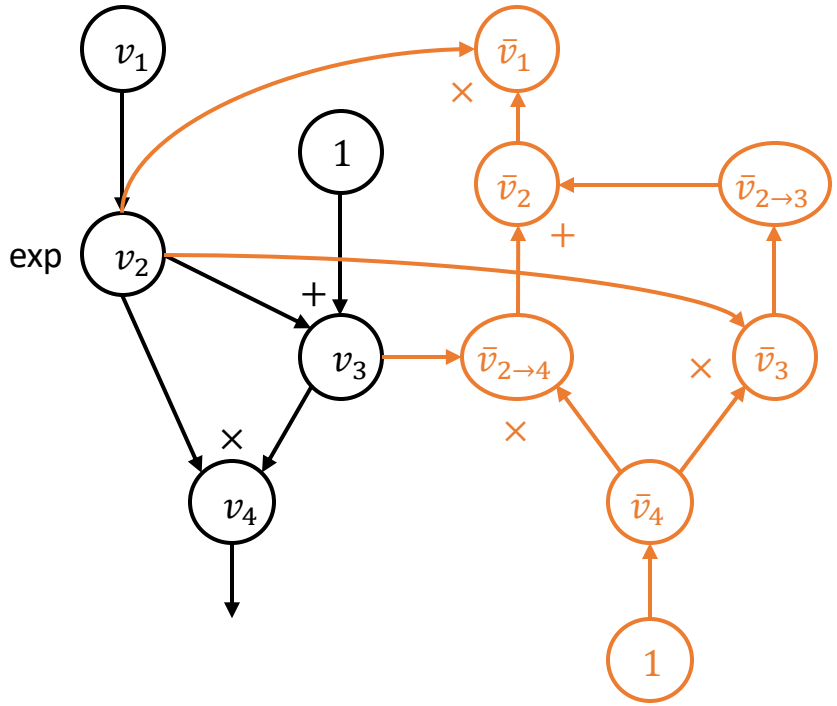
上海创智学院 Shanghai Innovation Institute

Reverse mode AutoDiff by computing in-place



- Run backward operations the same forward graph
- Used in earlier DL frameworks (caffe etc.)

Reverse mode AutoDiff by extending computational graph



- Construct separate graph nodes for adjoints.
- Used by modern deep learning frameworks.

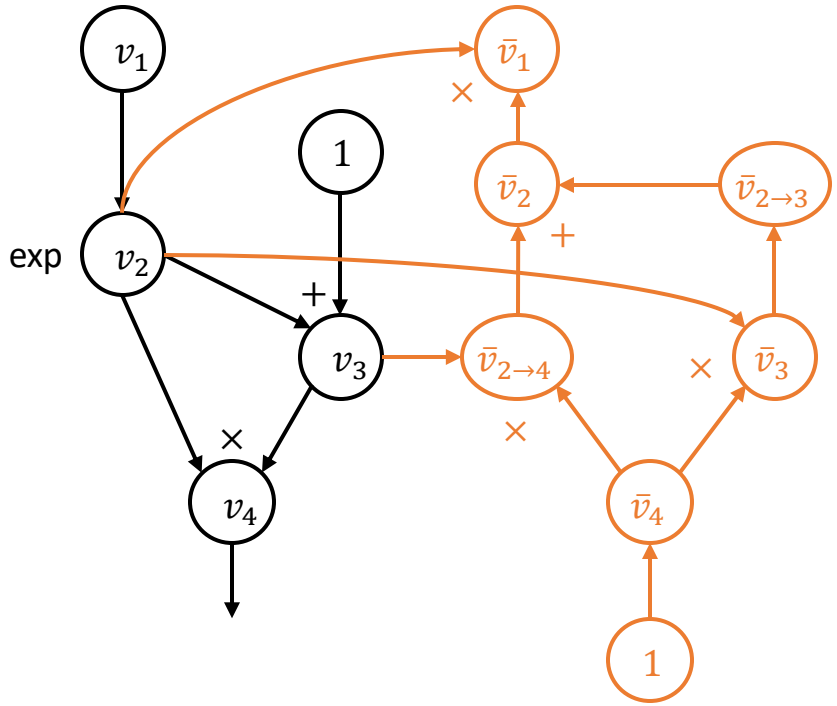
Why Extend Computational Graph

上海创智学院
Shanghai Innovation Institute

- Unified execution runtime

- Native handling Gradient of Gradient
  - The result of reverse mode AD is still a computational graph
  - We can extend that graph further by composing more operations and run reverse mode AD again on the gradient

Reverse mode AutoDiff by extending computational graph



- Construct separate graph nodes for adjoints.
- Used by modern deep learning frameworks.

Acknowledgement

The development of this course, including its structure, content, and accompanying presentation slides, has been significantly influenced and inspired by the excellent work of instructors and institutions who have shared their materials openly. We wish to extend our sincere acknowledgement and gratitude to the following courses, which served as invaluable references and a source of pedagogical inspiration:

- Machine Learning Systems[15-442/15-642], by **Tianqi Chen** and **Zhihao Jia** at **CMU**.
- Advanced Topics in Machine Learning (Systems)[CS6216], by **Yao Lu** at **NUS**

While these materials provided a foundational blueprint and a wealth of insightful examples, all content herein has been adapted, modified, and curated to meet the specific learning objectives of our curriculum. Any errors, omissions, or shortcomings found in these course materials are entirely our own responsibility. We are profoundly grateful for the contributions of the educators listed above, whose dedication to teaching and knowledge-sharing has made the creation of this course possible.

System for Artificial Intelligence

Thanks

Siyuan Feng
Shanghai Innovation Institute