Automatic Differentiation

Numerical Differentiation

从数学角度看，计算梯度的方法：

$$\frac{\partial f(\theta)}{\partial \theta_i}=\lim_{\epsilon\to 0}\frac{f(\theta+\epsilon e_i)-f(\theta)}{\epsilon} \tag{2}$$

精度更高的梯度算法：

$$\frac{\partial f(\theta)}{\partial \theta_i}=\frac{f(\theta+\epsilon e_i)-f(\theta-\epsilon e_i)}{2\epsilon}+o(\epsilon^2) \tag{3}$$

> 由泰勒展开即可得到

缺陷：不够精确；要算 2n 次 f, 效率低

Numerical Gradient Checking

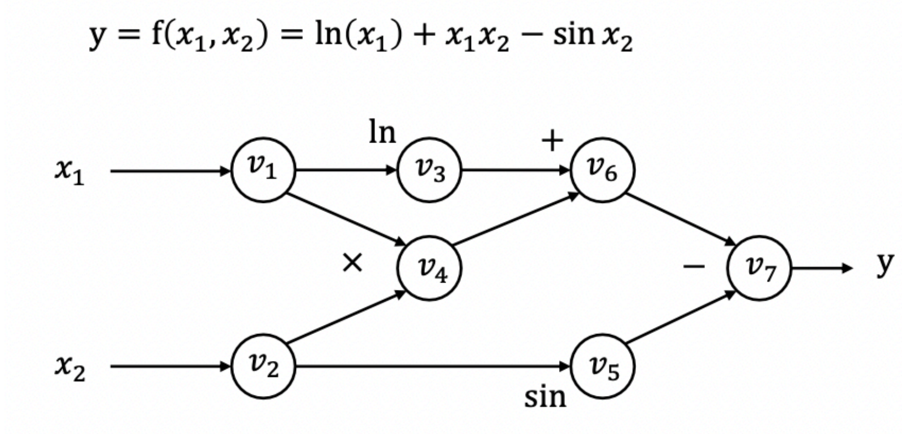
数学方法可以用于测试实现的自微分算法是否准确

$$\delta^T\nabla_\theta f(\theta)=\frac{f(\theta+\epsilon\delta)-f(\theta-\epsilon\delta)}{2\epsilon}+o(\epsilon^2) \tag{4}$$

其中 δ 是单位向量，$\nabla_\theta f(\theta)$ 由自微分算法得出。

Computational Graph

机器学习框架的核心，是一个用于表示计算流程的 DAG



DAG 的每个节点代表中间计算结果，边代表计算间的输入输出关系，按拓扑顺序进行计算

Forward Mode Automatic Differentiation (AD)

核心思想：从输入开始向后计算对每个输入分量的微分

定义 $\dot{v_i} = \frac{\partial v_i}{\partial x_1}$ (x_1 是某一个输入分量)

可按照计算图的拓扑顺序来迭代求出 $\dot{v_i}$

例如，对于上述计算图有：

Forward AD trace

$$\dot{v_1} = 1$$
$$\dot{v_2} = 0$$
$$\dot{v_3} = \dot{v_1}/v_1 = 0.5$$
$$\dot{v_4} = \dot{v_1}v_2 + \dot{v_2}v_1 = 1\times 5 + 0\times 2 = 5$$
$$\dot{v_5} = \dot{v_2}\cos v_2 = 0\times \cos 5 = 0$$
$$\dot{v_6} = \dot{v_3} + \dot{v_4} = 0.5 + 5 = 5.5$$
$$\dot{v_7} = \dot{v_6} - \dot{v_5} = 5.5 - 0 = 5.5$$

Now we have $\frac{\partial y}{\partial x_1} = \dot{v_7} = 5.5$

Limitation of Forward Mode AD

Forward mode AD 的缺点在于计算次数取决于输入个数，对于 $f: R^n \to R^k$, 需要 n 次 forward AD pass 来得到对所有输入分量的微分。而在 ML 场景下，通常 k = 1, 而 n 很大。

因此，引入 Reverse mode AD.

Reverse Mode AD

核心思想：从输出开始往回计算对每个输入分量的微分

定义 **adjoint**: $\overline{v_i} = \frac{\partial y}{\partial v_i}$

可按照计算图的反向拓扑顺序来迭代求出 $\overline{v_i}$

例如，对于上述计算图有：

Reverse AD evaluation trace

$$
\begin{aligned}
\overline{v_7} &= \frac{\partial y}{\partial v_7} = 1 \\
\overline{v_6} &= \overline{v_7}\frac{\partial v_7}{\partial v_6} = \overline{v_7}\times 1 = 1 \\
\overline{v_5} &= \overline{v_7}\frac{\partial v_7}{\partial v_5} = \overline{v_7}\times(-1) = -1 \\
\overline{v_4} &= \overline{v_6}\frac{\partial v_6}{\partial v_4} = \overline{v_6}\times 1 = 1 \\
\overline{v_3} &= \overline{v_6}\frac{\partial v_6}{\partial v_3} = \overline{v_6}\times 1 = 1 \\
\overline{v_2} &= \overline{v_5}\frac{\partial v_5}{\partial v_2} + \overline{v_4}\frac{\partial v_4}{\partial v_2} = \overline{v_5}\times \cos v_2 + \overline{v_4}\times v_1 = -0.284 + 2 = 1.716 \\
\overline{v_1} &= \overline{v_4}\frac{\partial v_4}{\partial v_1} + \overline{v_3}\frac{\partial v_3}{\partial v_1} = \overline{v_4}\times v_2 + \overline{v_3}\frac{1}{v_1} = 5 + \frac{1}{2} = 5.5
\end{aligned}
$$

特别地，v_2 被作为两个节点的输入，此时可将 y 看作 $f(v_4, v_5)$, 其中v_4, v_5 均为 v_2 的函数，进而有

$$
\begin{aligned}
\overline{v_2} &= \frac{\partial y}{\partial v_2} = \frac{\partial f(v_4, v_5)}{\partial v_4} \cdot \frac{\partial v_4}{\partial v_2} + \frac{\partial f(v_4, v_5)}{\partial v_5} \cdot \frac{\partial v_5}{\partial v_2} \\
&= \overline{v_4} \cdot \frac{\partial v_4}{\partial v_2} + \overline{v_5} \cdot \frac{\partial v_5}{\partial v_2}
\end{aligned} \tag{5}
$$

因此定义 **partial adjoint**: $\overline{v_{i\to j}} = \overline{v_j} \cdot \frac{\partial v_j}{\partial v_i}$

于是有

$$
\overline{v_i} = \sum_{j \in next(i)} \overline{v_{i\to j}} \tag{6}
$$

Reverse AD Algorithm

```
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 ∈ 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
```

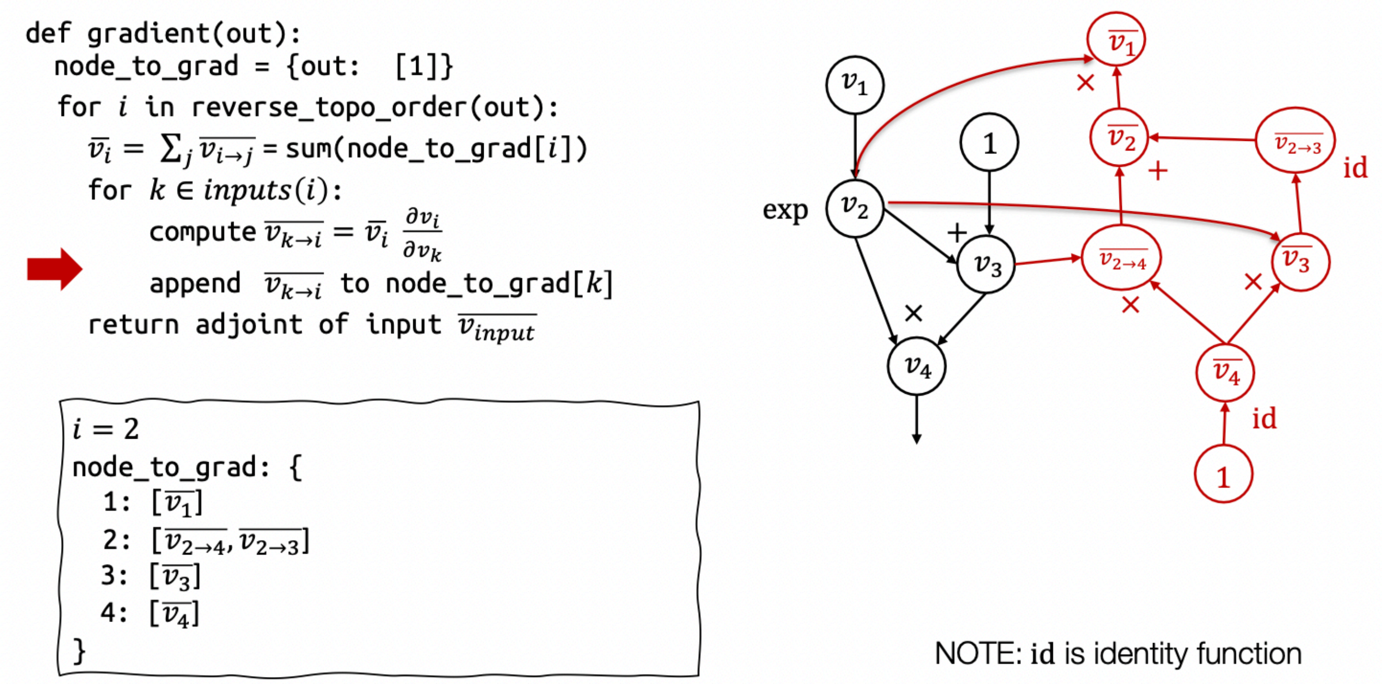
Dictionary that records a list of partial adjoints of each node

Sum up partial adjoints

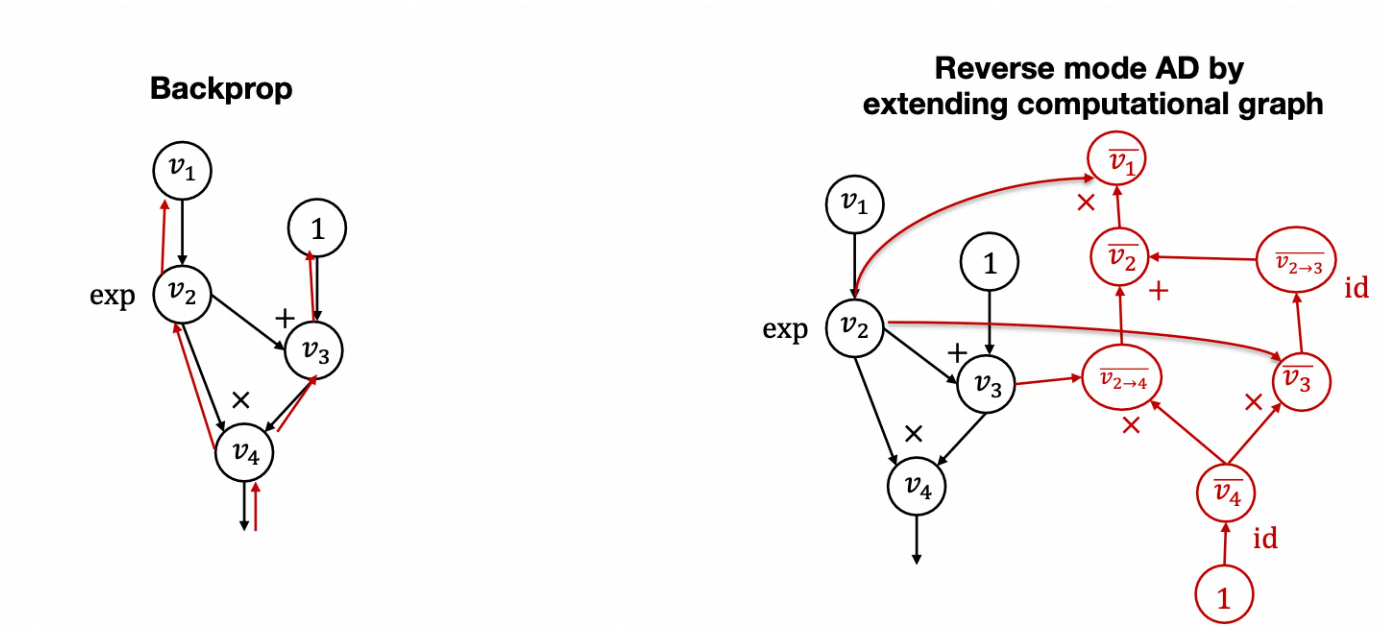
"Propagates" partial adjoint to its input

Extend Computational Graph

Reverse mode AD 生成了新的计算图，详细过程见 slide



Reverse Mode AD vs Backprop



Backprop 是在同一个图上进行计算，而 Reverse mode AD 对计算图进行了扩展，新图的节点是 adjoint.

Reverse Mode AD 可以轻松地计算梯度的梯度

Reverse Mode AD 可以对扩展图进行更好的优化，不需要保证对称性，提高效率

Reverse Mode AD on Tensors

Define adjoint for tensor values $\bar{Z} = \begin{bmatrix} \frac{\partial y}{\partial Z_{1,1}} & \dots & \frac{\partial y}{\partial Z_{1,n}} \\ \dots & \dots & \dots \\ \frac{\partial y}{\partial Z_{m,1}} & \dots & \frac{\partial y}{\partial Z_{m,n}} \end{bmatrix}$

设 $Z_{ij} = \sum_k X_{ik}W_{kj}, v = f(Z)$

即 $Z = XW, v = f(Z)$

则有

$$\overline{X_{i,k}} = \sum_j \frac{\partial Z_{i,j}}{\partial X_{i,k}} \overline{Z_{i,j}} = \sum_j W_{k,j} \overline{Z_{i,j}} \tag{7}$$

即

$$\overline{X} = \overline{Z} W^T \tag{8}$$

Reverse Mode AD on Data Structures

Takeaway: 定义 adjoint 通常与前向值和 adjoint 反向传播规则使用相同的数据类型