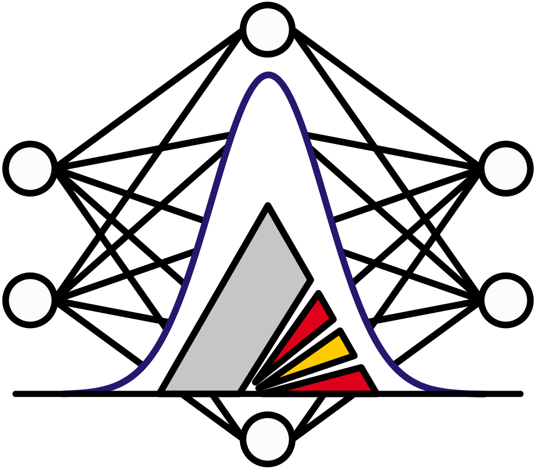


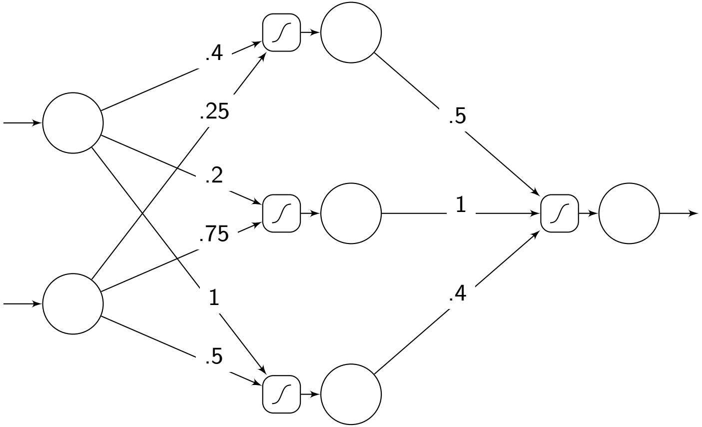
Beyond Backpropagation: Automatic Differentiation

Matthias Richter

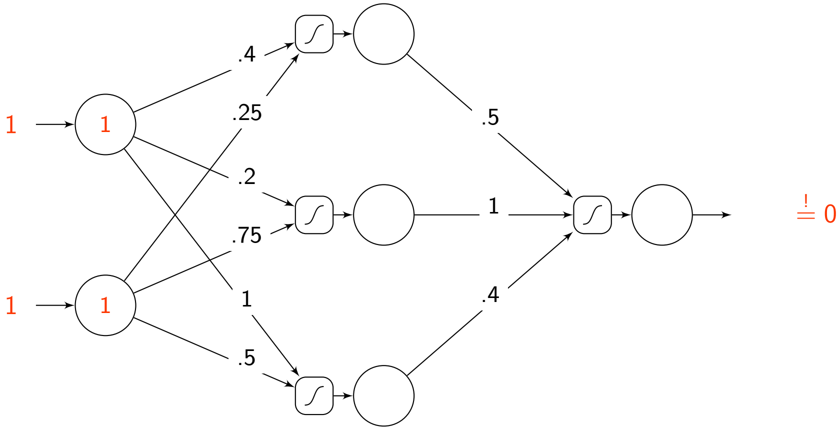
matthias.richter@kit.edu

@the_vrld vrld vrld (also on Researchgate)

Backpropagation!

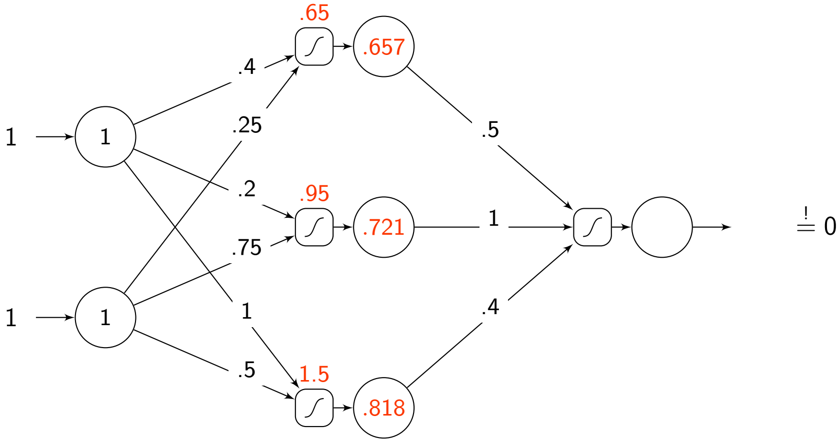


Backpropagation!



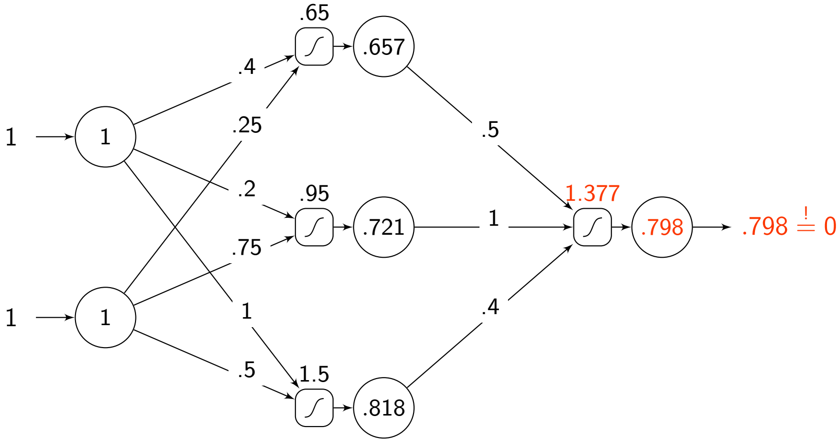
Forward pass: Transform input using model parameters

Backpropagation!



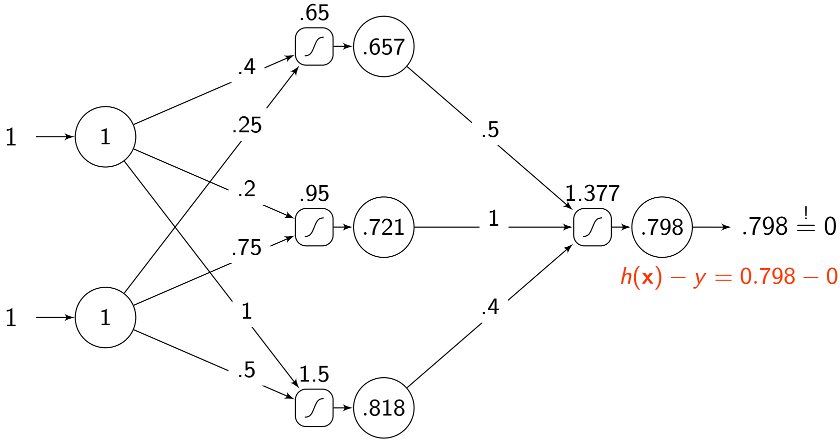
Forward pass: Transform input using model parameters

Backpropagation!



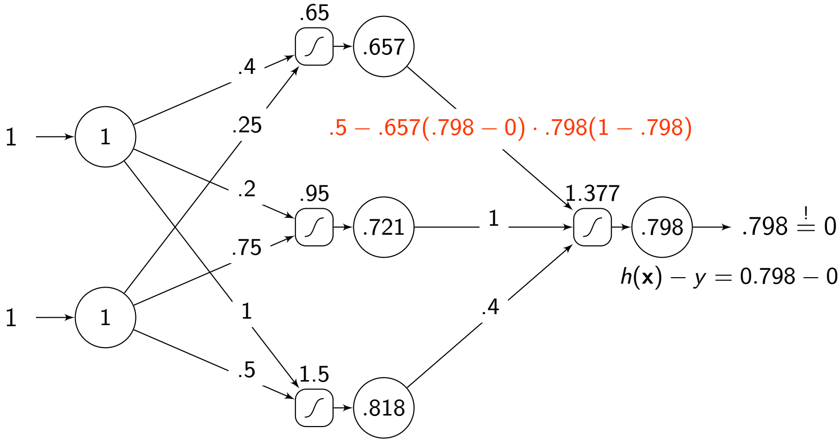
Forward pass: Transform input using model parameters

Backpropagation!



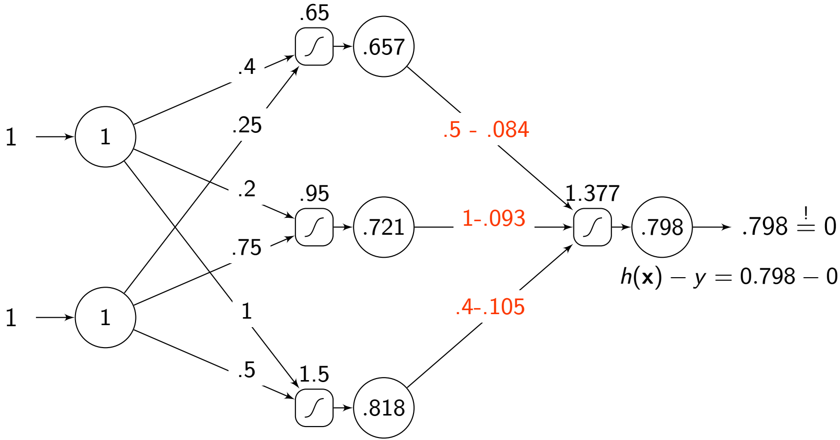
Backward pass: Adjust parameters to minimize error

Backpropagation!



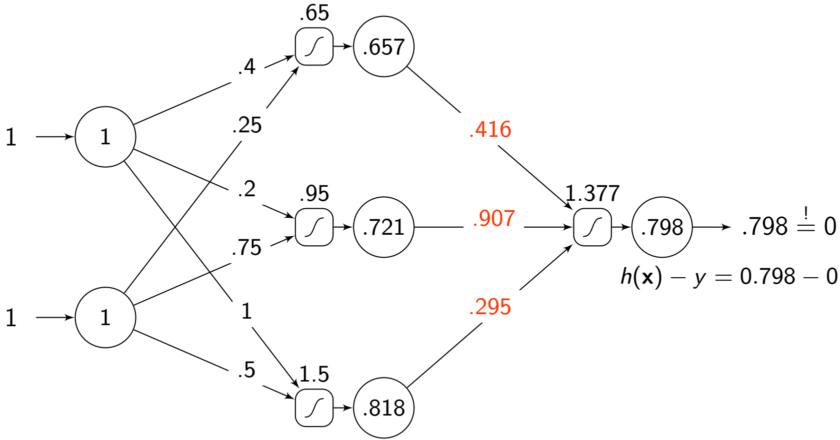
Backward pass: Adjust parameters to minimize error

Backpropagation!



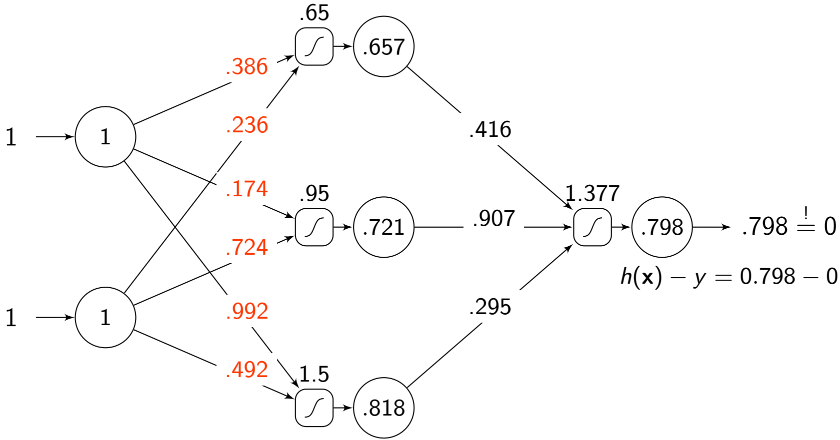
Backward pass: Adjust parameters to minimize error

Backpropagation!



Backward pass: Adjust parameters to minimize error

Backpropagation!



Backward pass: Adjust parameters to minimize error

Backpropagation?!

$$\Delta w_{ik} = -\eta \frac{\partial E}{\partial w_{ik}} = -\eta \begin{cases} o_i(o_k - y_k)o_k(1-o_k) & \text{if } k \text{ is output} \\ o_i\left(\sum_l \delta_l w_{kl}\right)o_k(1-o_k) & \text{else} \end{cases}$$

where

$$\delta_k = \frac{\partial E}{\partial o_k}\frac{\partial o_k}{\partial \text{net}_k} = \begin{cases} (o_k - y_k)o_k(1-o_k) & \text{if } k \text{ is output} \\ \left(\sum_l \delta_l w_{kl}\right)o_k(1-o_k) & \text{else} \end{cases}$$

w_{ik} ... weight from neuron i to neuron k

η ... learning rate

o_i ... output of neuron i

y_k ... desired output of neuron k

ϕ ... activation function

net_k ... input to neuron k

Backpropagation?!

$$\Delta w_{ik} = -\eta \frac{\partial E}{\partial w_{ik}} = -\eta \begin{cases} o_i(o_k - y_k)o_k(1 - o_k) & \text{if } k \text{ is output} \\ o_i\left(\sum_l \delta_l w_{kl}\right) o_k(1 - o_k) & \text{else} \end{cases}$$

where

$$\delta_k = \frac{\partial E}{\partial o_k}\frac{\partial o_k}{\partial \mathrm{net}_k} = \begin{cases} (o_k - y_k)o_k(1 - o_k) & \text{if } k \text{ is output} \\ \left(\sum_l \delta_l w_{kl}\right) o_k(1 - o_k) & \text{else} \end{cases}$$

w_{ik} ... weight from neuron i to neuron k

η ... learning rate

o_i ... output of neuron i

y_k ... desired output of neuron k

ϕ ... activation function

net_k ... input to neuron k

Backpropagation is gradient descent

$$\mathbf{w}_t = \mathbf{w}_{t-1} - \eta \Delta \mathbf{w}_t \quad \text{where} \quad \Delta \mathbf{w}_t = \frac{\partial}{\partial \mathbf{w}} \left(\sum_i \| h_{\mathbf{w}_t}(\mathbf{x}_i) - \mathbf{y}_i \|^2 \right)$$

$\mathbf{w}_t$ ... Parameters of the net

$\mathbf{x}_i$ ... i-th training sample

$\mathbf{y}_i$ ... i-th ground truth

$h(\mathbf{x})$ ... Output of the net (vectorial)

$$\boxed{t \to \infty \;\Rightarrow\; E = \sum_i \| h_{\mathbf{w}_\infty}(\mathbf{x}_i) - \mathbf{y}_i \|^2 \quad \text{(locally) minimal}}$$

Backpropagation is gradient descent

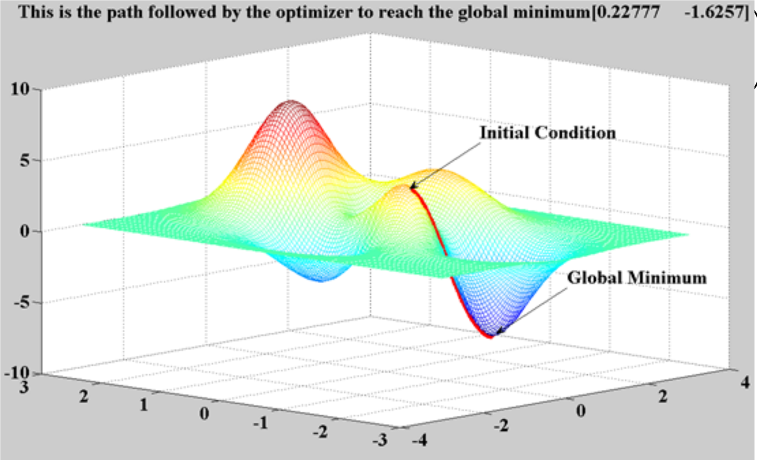
w

w

x

y

$h($x



Source: mathworks.com/matlabcentral/fileexchange/27631

Learning is optimization

$$\mathbf{w}^\star = \arg\min_{\mathbf{w}} \mathcal{L}(\mathbf{w})$$

- Backpropagation:

$$\mathcal{L}(\mathbf{w}) = \sum \| h_{\mathbf{w}}(\mathbf{x}_i) - \mathbf{y}_i \|^2$$

- Support Vector Machine:

$$\mathcal{L}(\mathbf{w}) = \|\mathbf{w}\|^2 + C \sum \max\left(0, 1 - y_i \cdot \left(\mathbf{w}^\top \phi(\mathbf{x}_i) + b\right)\right)$$

- Maximum Likelihood:

$$\mathcal{L}(\mathbf{w}) = -\prod p(\mathbf{x}_i | \mathbf{w})$$

How to optimize

$$\mathbf{w}^\star = \arg\min_{\mathbf{w}} f(\mathbf{w})$$

Pen & Paper

$$\nabla f(\mathbf{w}^\star) \overset{!}{=} 0$$

- Not always possible
- Can be tedious

How to optimize

$$\mathbf{w}^\star = \arg\min_{\mathbf{w}} f(\mathbf{w})$$

Pen & Paper

$$\nabla f(\mathbf{w}^\star) \stackrel{!}{=} 0$$

- Not always possible
- Can be tedious

Algorithmic / Iterative

$$\mathbf{w}_{t+1} = \mathbf{w}_t + \Delta\mathbf{w}_t$$

- Gradient descent

  $$\Delta\mathbf{w}_t = -\eta\,\nabla f(\mathbf{w}_t)$$

- Newton-Raphson

  $$\Delta\mathbf{w}_t = -\eta\,(\mathbf{H}_f(\mathbf{w}_t))^{-1}\nabla f(\mathbf{w}_t)$$

- ...

How to optimize

$$\mathbf{w}^\star = \arg\min_{\mathbf{w}} f(\mathbf{w})$$

Pen & Paper

$$\nabla f(\mathbf{w}^\star) \stackrel{!}{=} 0$$

- Not always possible
- Can be tedious

Algorithmic / Iterative

$$\mathbf{w}_{t+1} = \mathbf{w}_t + \Delta\mathbf{w}_t$$

- Gradient descent

$$\Delta\mathbf{w}_t = -\eta \nabla f(\mathbf{w}_t)$$

- Newton-Raphson

$$\Delta\mathbf{w}_t = -\eta \left(\mathbf{H}_f(\mathbf{w}_t)\right)^{-1} \nabla f(\mathbf{w}_t)$$

- ...

Solve by thinking

Target formula

$$l_{t+1} = 4l_t(1-l_t) \qquad \text{where} \qquad l_1 = x$$
$$f(x) = l_4 = 64x(1-x)(1-2x)^2 \cdot (1-8x+8x^2)^2$$

Derive using brain

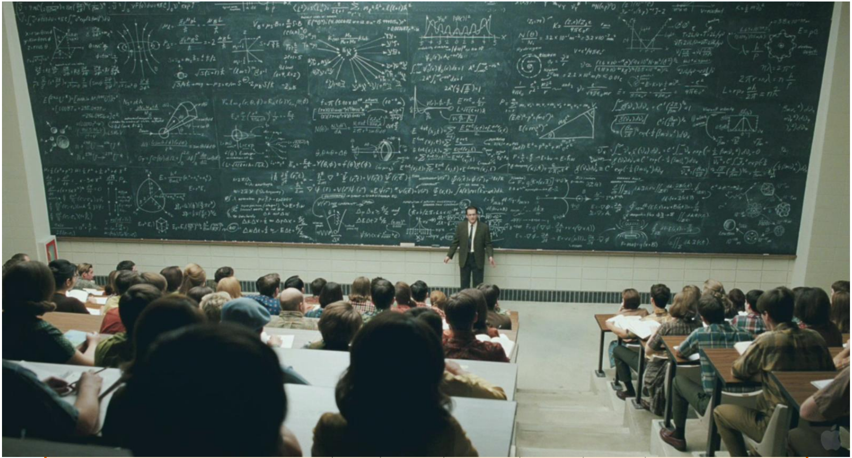
$$\begin{aligned} f'(x) = &128x(1-x)(16x-8)(1-2x)^2(1-8x+8x^2) \\ &+ 64(1-x)(1-2x)^2(1-8x+8x^2)^2 - \dots \end{aligned}$$

Code using brain

```
function df(x)
    return 128x * (1-x) * (16x-8) * (1-2x)^2
            * (1-8x+8x^2) + ...
end
```

Solve by thinking

Target formula



```
                * (1-8x+8x^2) + ...
end
```

Numeric differentiation

Target formula

$$l_{t+1} = 4l_t(1 - l_t) \qquad \text{where} \qquad l_1 = x$$
$$f(x) = l_4$$

Code using brain

```
function f(x)
    x = 4x * (1-x) for i = 1:4
    return x
end
```

Numeric approximation

```
function df(x)
    return (f(x+h) - f(x)) / h
end
```

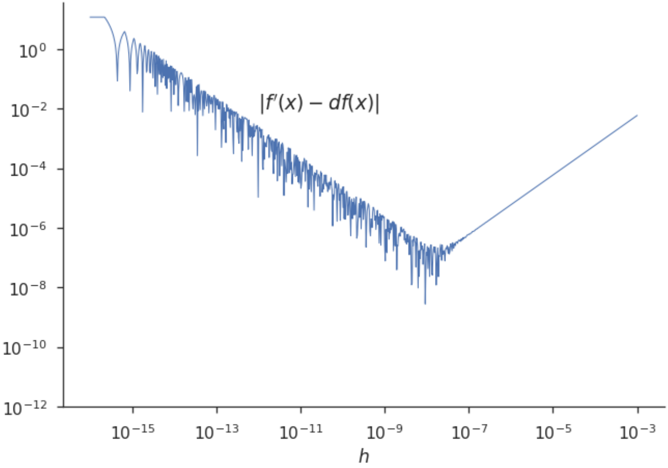
Numeric differentiation

$$\frac{\partial}{\partial x_i} f(\mathbf{x}) \approx \frac{f(\mathbf{x} + h\,\mathbf{e}_i) - f(\mathbf{x})}{h}$$

- $h \to 0$: Textbook definition of the derivative
- Trivial to implement
- Inefficient:
  - $2d$ evaluations of $f(\mathbf{x})$ for $\nabla f(\mathbf{x})$ for $\mathbf{x} \in \mathbb{R}^d$
  - $4d^2$ evaluations for $\mathbf{H}_f(\mathbf{x})$ for $\mathbf{x} \in \mathbb{R}^d$
- Unstable:
  - Truncation error ($h \neq 0$)
  - Round-off errors (IEEE754)

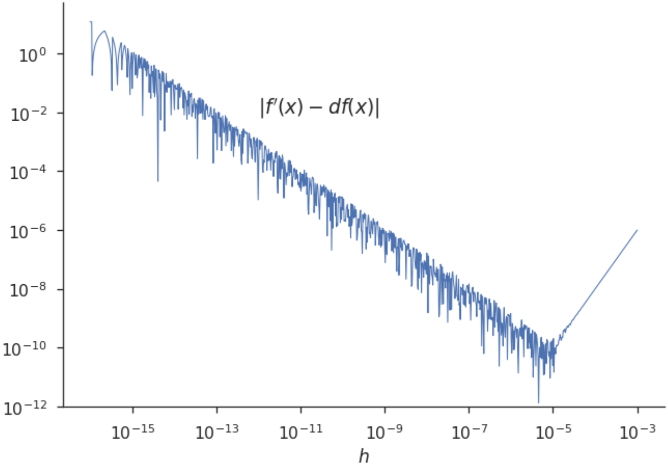
Numeric differentiation

$$\frac{\partial}{\partial x_i} f(\mathbf{x}) \approx \frac{f(\mathbf{x} + h\,\mathbf{e}_i) - f(\mathbf{x})}{h}$$



Numeric differentiation

$$\frac{\partial}{\partial x_i} f(\mathbf{x}) \approx \frac{f(\mathbf{x} + h\,\mathbf{e}_i) - f(\mathbf{x} - h\,\mathbf{e}_i)}{2h}$$



Symbolic differentiation / CAS

Target formula

$$l_{t+1} = 4l_t(1 - l_t) \qquad \text{where} \qquad l_1 = x$$
$$f(x) = l_4 = 64x(1-x)(1-2x)^2 \cdot (1-8x+8x^2)^2$$

Code using brain (closed form!)

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

Symbolic transformation

```
function df(x)
    return 128x * (1-x) * (16x-8) * (1-2x)^2
            * (1-8x+8x^2) + ...
end
```

Symbolic differentiation / CAS

$$\big(u(x)\,v(x)\big)' \mapsto u'(x)\,v(x) + u(x)\,v'(x)$$

- Symbolic transformations / derivative rules
- Exact solution
- Can look at the formula
- No branches, no loops, no fun
- *Expression swell*

$f(x)$	$d/dxf(x)$	$d/dxf(x)$ (shortened)
x	1	1
$4x(1-x)$	$4(1-x)-4x$	$4-8x$
$16x(1-x)(1-2x)^2$	$16(1-x)(1-2x)^2$ $-16x(1-2x)^2$ $-64x(1-x)(1-2x)$	$16(1-10x+24x^2-16x^3)$

Beyond: Automatic differentiation

Target formula

$$
\begin{aligned}
l_{t+1} &= 4l_t(1 - l_t) \qquad \text{where} \qquad l_1 = x \\
f(x) &= l_4
\end{aligned}
$$

Code using brain

```
function f(x)
    x = 4x * (1-x) for i = 1:4
    return x
end
```

Automatic differentiation

```
function df(x)
    return gradient(f, x)
end
```

Differentiation

Every calculation is a **combination of basic operations**

- Define basic operations and their derivatives

$$\begin{aligned} u(x_0) + v(x_0) &\rightsquigarrow u'(x_0) + v'(x_0) \\ u(x_0)\,v(x_0) &\rightsquigarrow u'(x_0)\,v(x_0) + u(x_0)\,v'(x_0) \\ \exp(x_0) &\rightsquigarrow \exp(x_0) \\ \dots \end{aligned}$$

- Use chain rule for everything else

$$(f \circ g)'(x_0) \rightsquigarrow f'(g(x_0))\,g'(x_0)$$

- **No thinking required!**

A new old thing

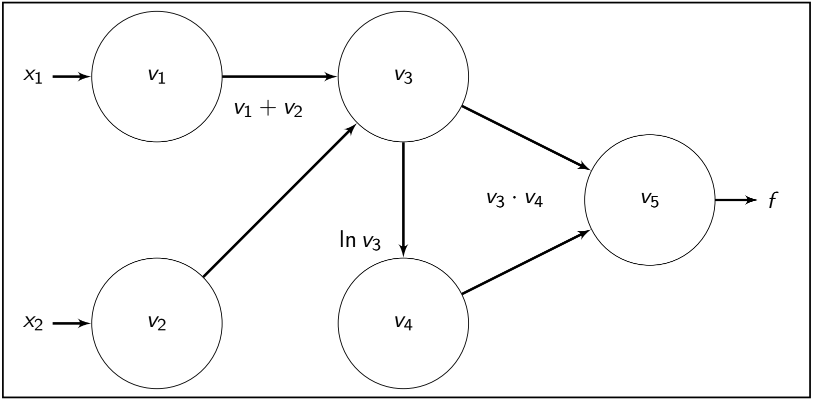
Early work

- J.F. Nolan: **Analytical Differentiation on a Digital Computer**, *Master Thesis, MIT*, 1953
- H.G. Kahrimanian: **Analytical Differentiation by a Digital Computer**, *Master Thesis, Temple University*, 1953
- R. Wengert: **A Simple Automatic Derivative Evaluation Program**, *Communiations of the ACM*, 1964

. . . all this and more at

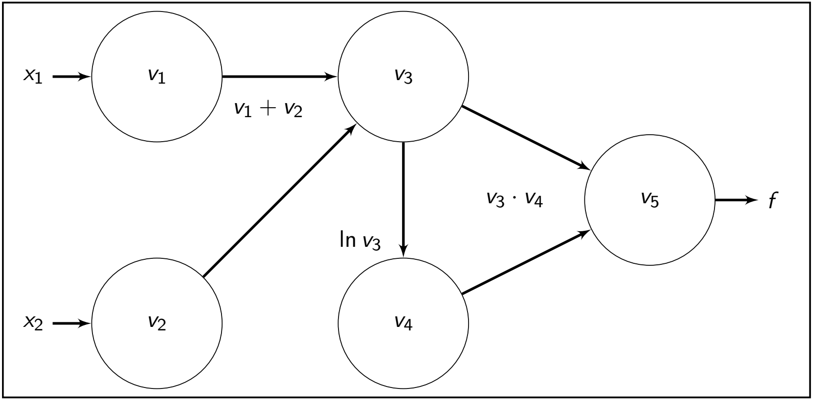
`www.autodiff.org`

Graph representation



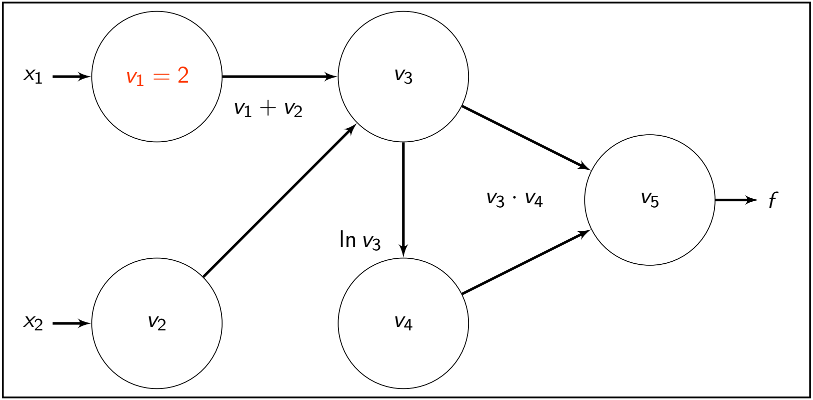
$$f(\underbrace{x_1}_{v_1}, \underbrace{x_2}_{v_2}) = \underbrace{\underbrace{(x_1 + x_2)}_{v_3} \cdot \underbrace{\ln \underbrace{(x_1 + x_2)}_{v_3}}_{v_4 = \ln v_3}}_{v_5 = v_3 \cdot v_4}$$

Computing with the graph



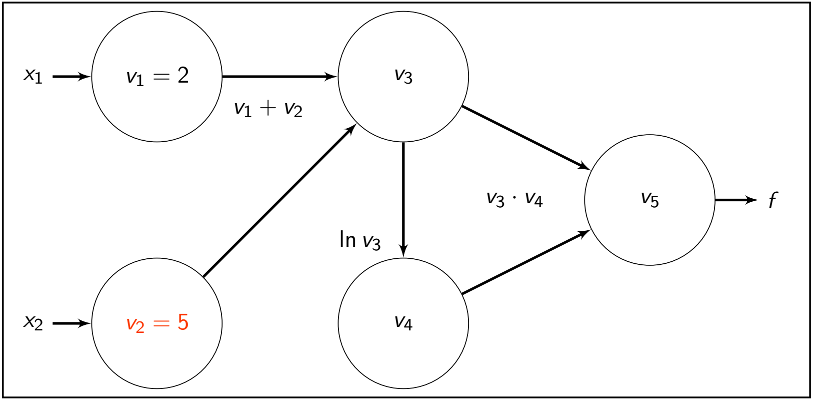
$$f(x_1, x_2) = (x_1 + x_2) \cdot \ln(x_1 + x_2) \quad \rightsquigarrow \quad f(2, 5) = ?$$

Computing with the graph



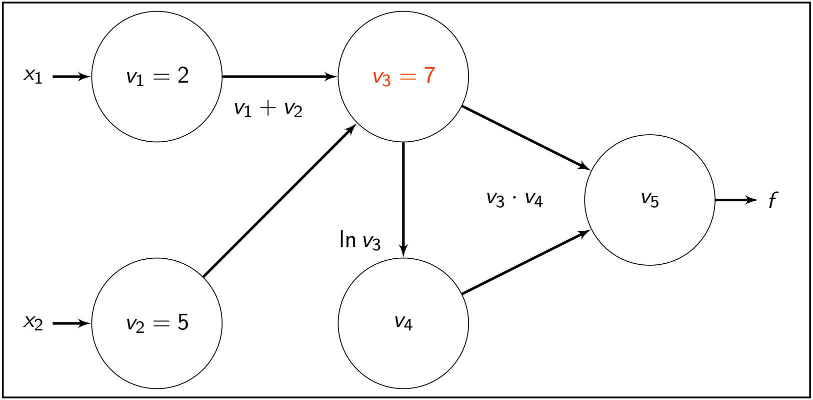
$$f(x_1, x_2) = (x_1 + x_2) \cdot \ln(x_1 + x_2) \quad \rightsquigarrow \quad f(2, 5) = ?$$

Computing with the graph



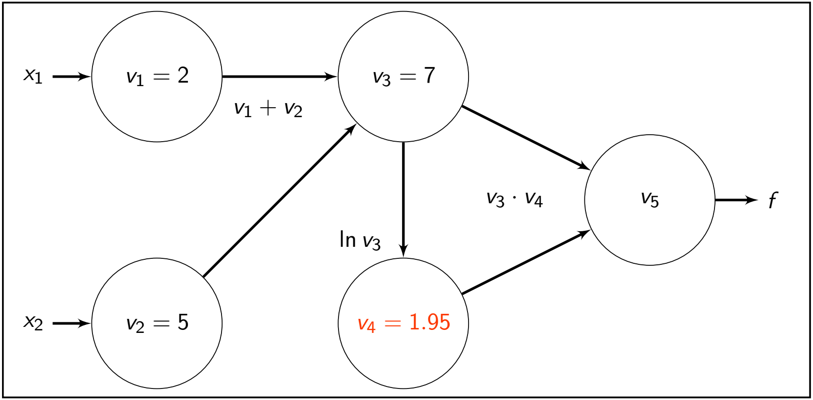
$$f(x_1, x_2) = (x_1 + x_2) \cdot \ln(x_1 + x_2) \quad \rightsquigarrow \quad f(2, 5) = ?$$

Computing with the graph



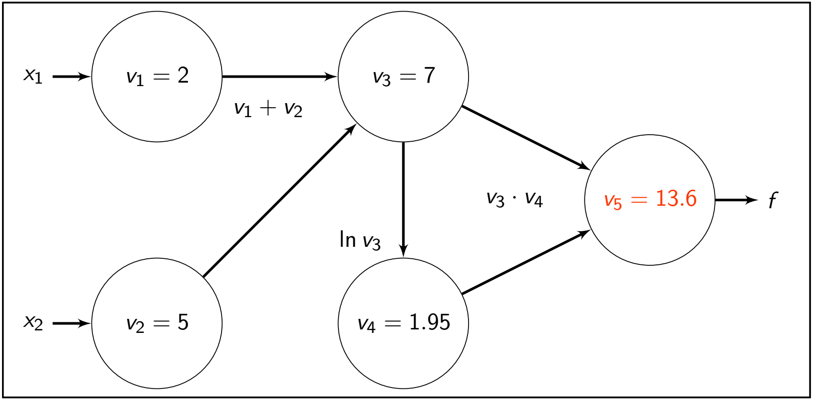
$$f(x_1, x_2) = (x_1 + x_2) \cdot \ln(x_1 + x_2) \quad \rightsquigarrow \quad f(2,5) = ?$$

Computing with the graph



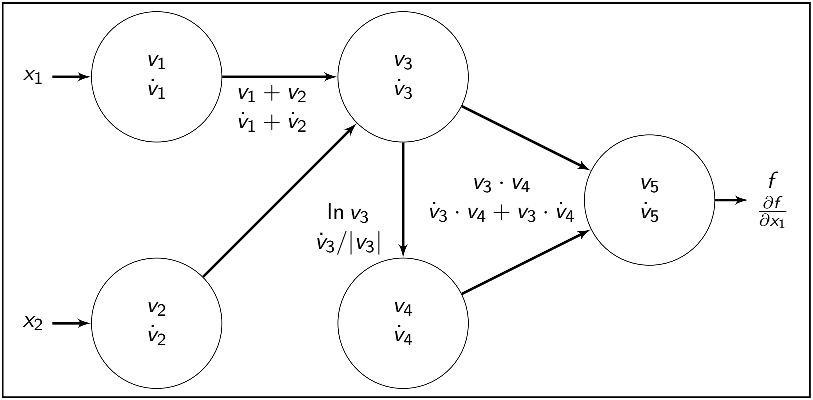
$$f(x_1, x_2) = (x_1 + x_2) \cdot \ln(x_1 + x_2) \quad \rightsquigarrow \quad f(2, 5) = ?$$

Computing with the graph



$$f(x_1, x_2) = (x_1 + x_2) \cdot \ln(x_1 + x_2) \quad \rightsquigarrow \quad f(2,5) = 13.6$$

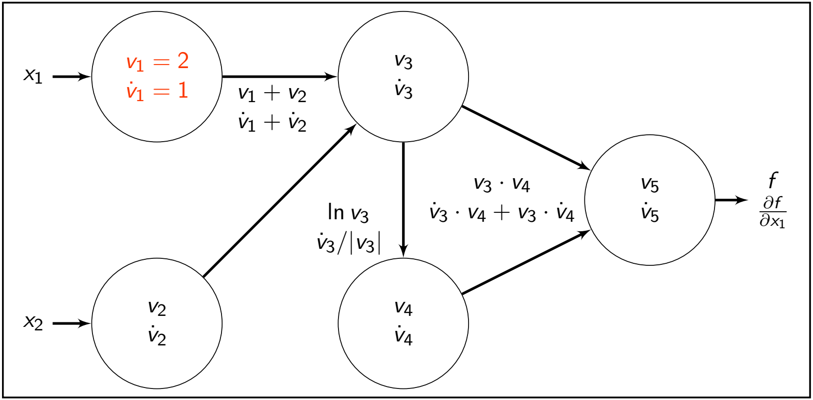
Forward mode AD



$$f(x_1, x_2) = (x_1 + x_2) \cdot \ln(x_1 + x_2) \quad \rightsquigarrow \quad \frac{\partial}{\partial x_1} f(2, 5) = ?$$

calculate $\dot{v}_i = \frac{\partial v_i}{\partial x_1}$ in every step

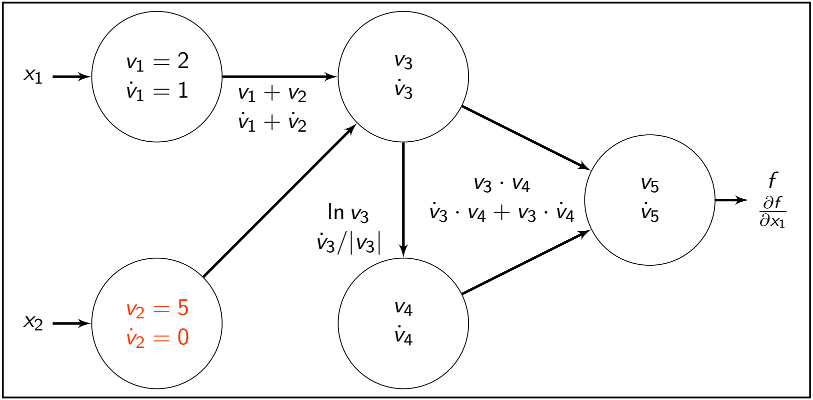
Forward mode AD



$$f(x_1, x_2) = (x_1 + x_2) \cdot \ln(x_1 + x_2) \quad \rightsquigarrow \quad \frac{\partial}{\partial x_1} f(2, 5) = ?$$

calculate $\dot{v}_i = \frac{\partial v_i}{\partial x_1}$ in every step

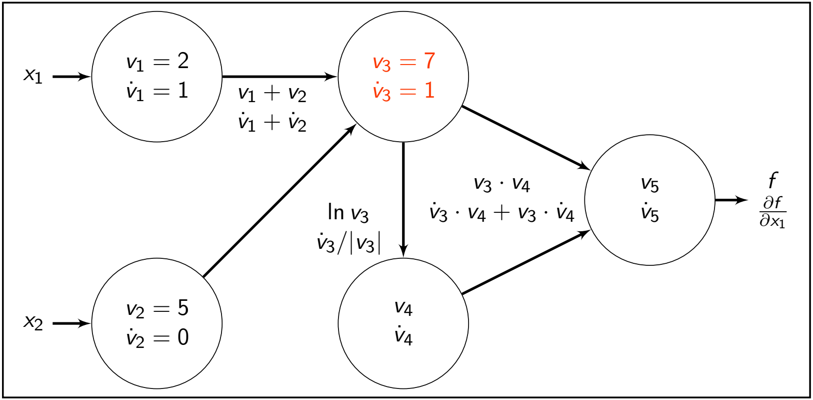
Forward mode AD



$$f(x_1, x_2) = (x_1 + x_2) \cdot \ln(x_1 + x_2) \quad \rightsquigarrow \quad \frac{\partial}{\partial x_1} f(2, 5) = ?$$

$$\boxed{\text{calculate } \dot{v}_i = \frac{\partial v_i}{\partial x_1} \text{ in every step}}$$

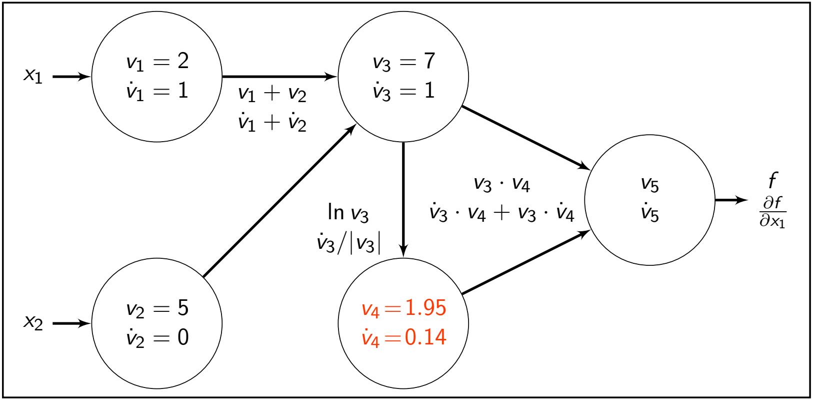
Forward mode AD



$$f(x_1, x_2) = (x_1 + x_2) \cdot \ln(x_1 + x_2) \quad \rightsquigarrow \quad \frac{\partial}{\partial x_1} f(2, 5) = ?$$

calculate $\dot{v}_i = \frac{\partial v_i}{\partial x_1}$ in every step

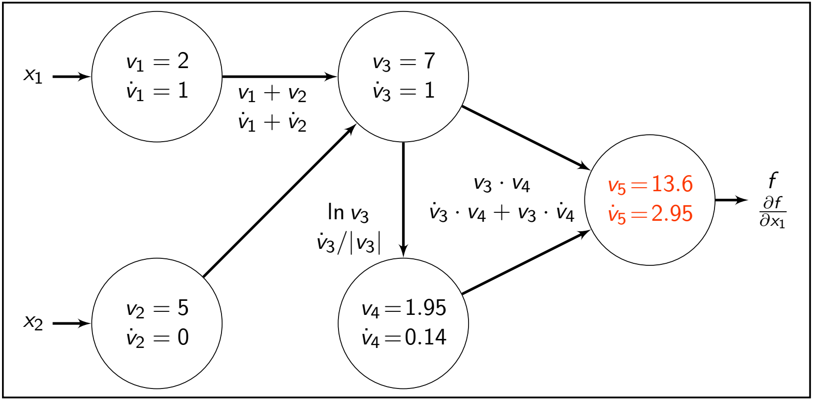
Forward mode AD



$$f(x_1, x_2) = (x_1 + x_2) \cdot \ln(x_1 + x_2) \quad \rightsquigarrow \quad \frac{\partial}{\partial x_1} f(2, 5) = ?$$

calculate $\dot{v}_i = \frac{\partial v_i}{\partial x_1}$ in every step

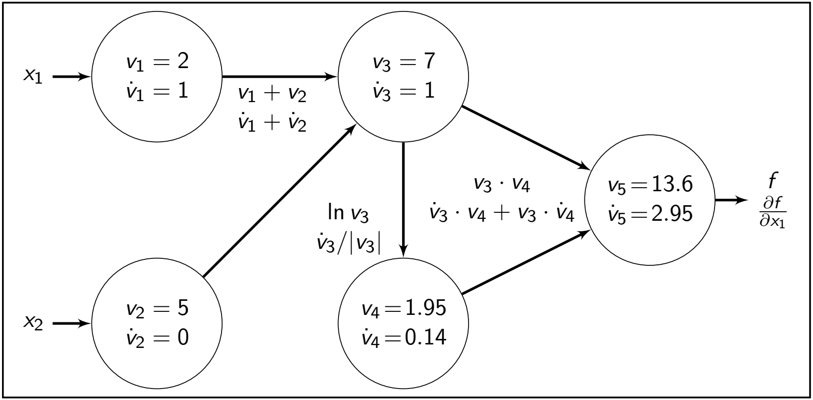
Forward mode AD



$$f(x_1, x_2) = (x_1 + x_2) \cdot \ln(x_1 + x_2) \quad \rightsquigarrow \quad \frac{\partial}{\partial x_1} f(2, 5) = ?$$

calculate $\dot{v}_i = \frac{\partial v_i}{\partial x_1}$ in every step

Forward mode AD



$$f(x_1, x_2) = (x_1 + x_2) \cdot \ln(x_1 + x_2) \quad \rightsquigarrow \quad \frac{\partial}{\partial x_1} f(2, 5) = 2.95$$

computes $\nabla_{\mathbf{x}} f(\mathbf{x}_0)$ exactly in d evaluations ($\mathbf{x} \in \mathbb{R}^d$)

Dual numbers

$$z = v + \dot{v}\epsilon \quad \text{where} \;\; v, \dot{v} \in \mathbb{R} \;\; \text{and} \;\; \epsilon^2 = 0$$

$$\begin{aligned}
\lambda\, z_u &= \lambda(u + \dot{u}\epsilon)) = \lambda\, u + \lambda\, \dot{u}\epsilon \\
z_u + z_v &= (u + \dot{u}\epsilon) + (v + \dot{v}\epsilon) = (u + v) + (\dot{u} + \dot{v})\epsilon \\
z_u \cdot z_v &= (u + \dot{u}\epsilon)\,(v + \dot{v}\epsilon) = uv + (\dot{u}v + \dot{v}u)\epsilon \\
&\dots
\end{aligned}$$

$\Rightarrow$ **Algebra reproduces derivative rules!**

Dual numbers

$$z = v + \dot{v}\epsilon \quad \text{where } \; v, \dot{v} \in \mathbb{R} \; \text{ and } \; \epsilon^2 = 0$$

$$\begin{aligned}
\lambda\, z_u &= \lambda(u + \dot{u}\epsilon)) = \lambda\, u + \lambda\, \dot{u}\epsilon \\
z_u + z_v &= (u + \dot{u}\epsilon) + (v + \dot{v}\epsilon) = (u + v) + (\dot{u} + \dot{v})\epsilon \\
z_u \cdot z_v &= (u + \dot{u}\epsilon)\,(v + \dot{v}\epsilon) = uv + (\dot{u}v + \dot{v}u)\epsilon \\
&\dots \\
f(v + \dot{v}\epsilon) &:= f(v) + f'(v)\dot{v}\epsilon \qquad \text{(defined)}
\end{aligned}$$

$\Rightarrow$ **Chain rule!**

$$\begin{aligned}
f(g(v + \dot{v}\epsilon)) &= f\left(g(v) + g'(v)\dot{v}\epsilon\right) \\
&= f\left(g(v)\right) + f'\left(g(v)\right) g'(v)\dot{v}\epsilon
\end{aligned}$$

Dual numbers

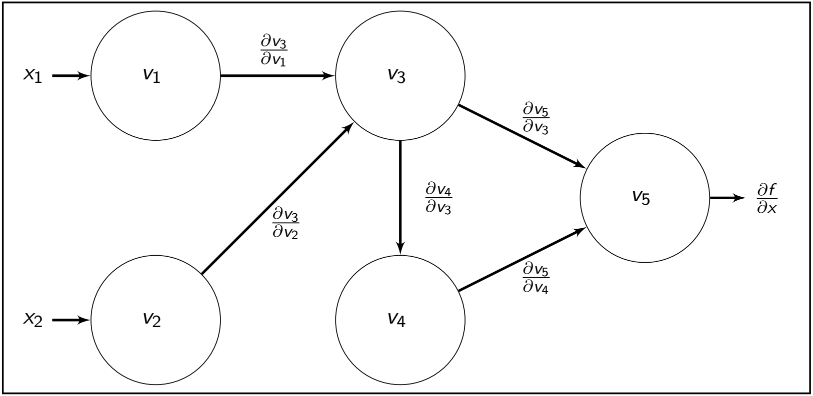
$$z = v + \dot{v}\epsilon \quad \text{where} \;\; v, \dot{v} \in \mathbb{R} \;\; \text{and} \;\; \epsilon^2 = 0$$

$$\begin{aligned}
\lambda\, z_u &= \lambda(u + \dot{u}\epsilon)) = \lambda\, u + \lambda\, \dot{u}\epsilon \\
z_u + z_v &= (u + \dot{u}\epsilon) + (v + \dot{v}\epsilon) = (u + v) + (\dot{u} + \dot{v})\epsilon \\
z_u \cdot z_v &= (u + \dot{u}\epsilon)\,(v + \dot{v}\epsilon) = uv + (\dot{u}v + \dot{v}u)\epsilon \\
&\ldots \\
f(v + \dot{v}\epsilon) &:= f(v) + f'(v)\dot{v}\epsilon \qquad\qquad \text{(defined)} \\
f(g(v + \dot{v}\epsilon)) &= f\,(g(v)) + f'\,(g(v))\,g'(v)\dot{v}\epsilon
\end{aligned}$$

Forward mode AD:

$$\frac{d}{dx} f(x_0) = \epsilon\text{-coefficient}\Big(f(x_0 + 1\epsilon)\Big)$$

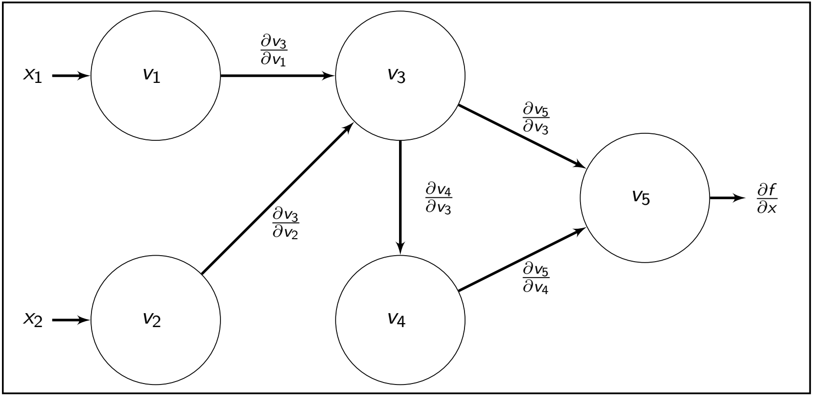
A shift in perspective



Generalized chain rule

$$\frac{\partial f}{\partial v_i} = \sum_{v_j \in \mathrm{Ch}(v_i)} \frac{\partial v_j}{\partial v_i} \cdot \frac{\partial f}{\partial v_j}$$

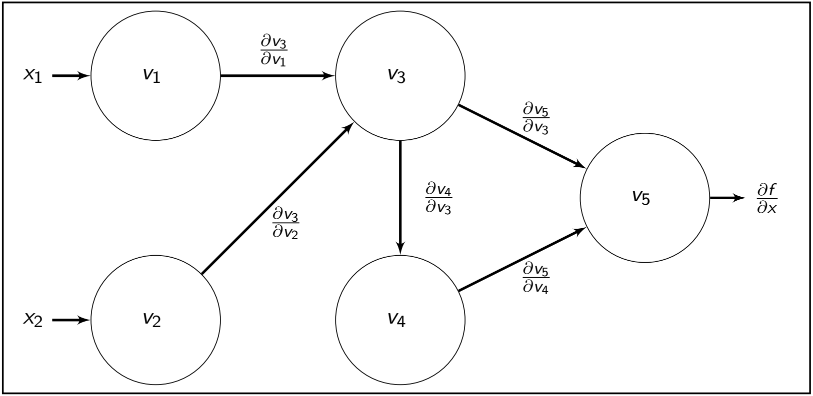
A shift in perspective



Generalized chain rule

$$\frac{\partial f}{\partial v_i} = \sum_{v_j \in \mathrm{Ch}(v_i)} \frac{\partial v_j}{\partial v_i} \cdot \frac{\partial f}{\partial v_j} = \sum_{v_j \in \mathrm{Ch}(v_i)} \frac{\partial v_j}{\partial v_i} \cdot \left(\sum_{v_k \in \mathrm{Ch}(v_j)} \frac{\partial v_k}{\partial v_j} \cdot \frac{\partial f}{\partial v_k} \right)$$

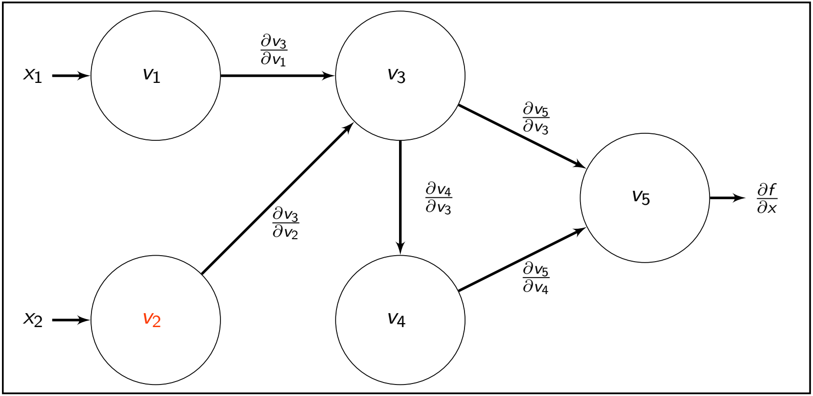
A shift in perspective



Apply chain rule to get gradient

$$\overline{v}_i := \frac{\partial f}{\partial v_i}$$

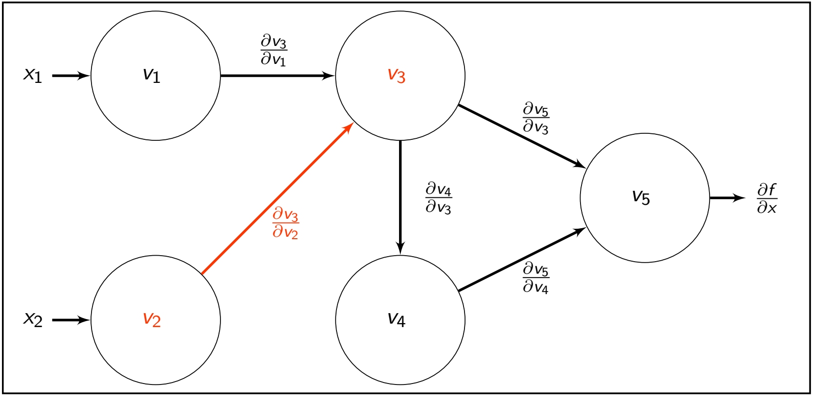
A shift in perspective



Apply chain rule to get gradient

$$\overline{v}_2 = \frac{\partial f}{\partial v_2}$$

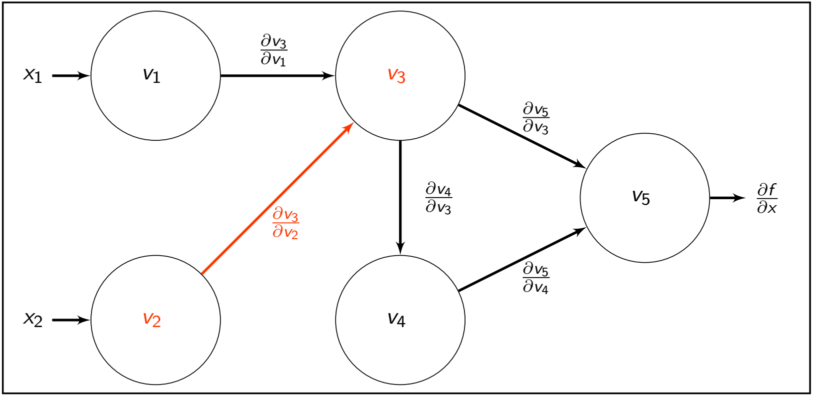
A shift in perspective



Apply chain rule to get gradient

$$\overline{v}_2 = \frac{\partial f}{\partial v_2} = \frac{\partial v_3}{\partial v_2}\frac{\partial f}{\partial v_3}$$

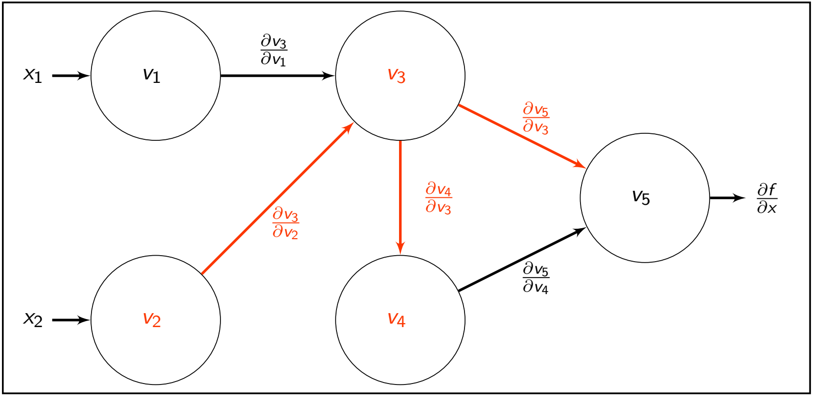
A shift in perspective



Apply chain rule to get gradient

$$\overline{v}_2 = \frac{\partial f}{\partial v_2} = \frac{\partial v_3}{\partial v_2} \overline{v}_3$$

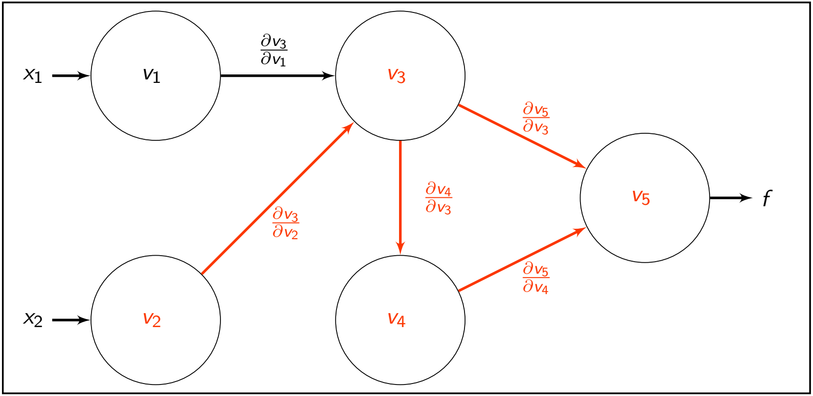
A shift in perspective



Apply chain rule to get gradient

$$\overline{v}_2 = \frac{\partial f}{\partial v_2} = \frac{\partial v_3}{\partial v_2}\left(\frac{\partial v_4}{\partial v_3}\overline{v}_4 + \frac{\partial v_5}{\partial v_3}\overline{v}_5\right)$$

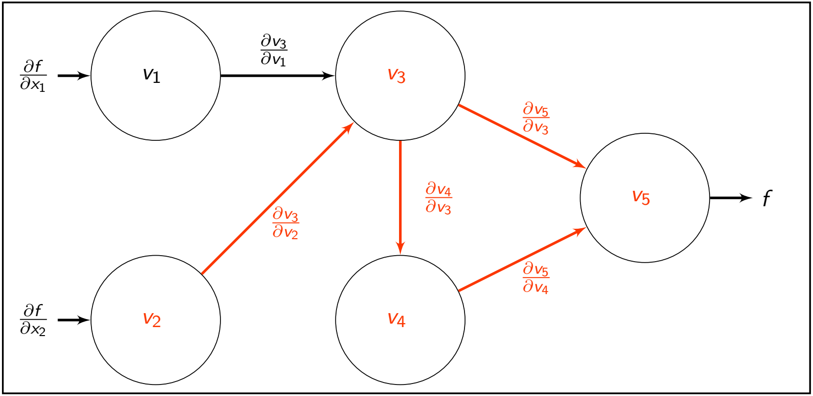
A shift in perspective



Apply chain rule to get gradient

$$\overline{v}_2 = \frac{\partial f}{\partial v_2} = \frac{\partial v_3}{\partial v_2}\left(\frac{\partial v_4}{\partial v_3}\left(\frac{\partial v_5}{\partial v_4}\overline{v}_5\right) + \frac{\partial v_5}{\partial v_3}\overline{v}_5\right)$$

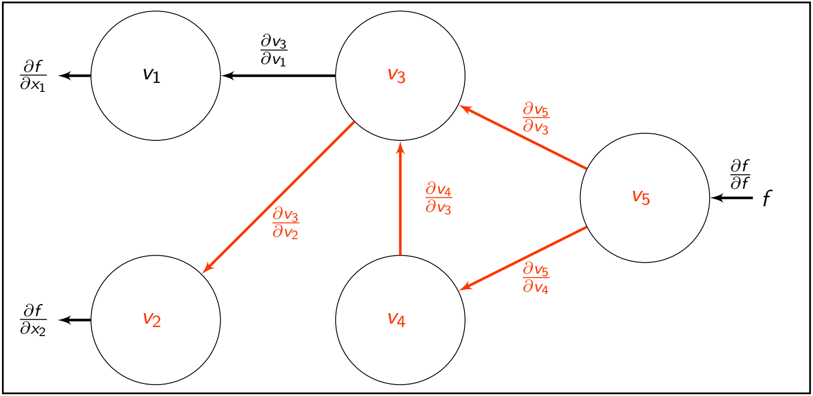
A shift in perspective



Apply chain rule to get gradient

$$\overline{v}_2 = \frac{\partial f}{\partial v_2} = \frac{\partial v_3}{\partial v_2}\left(\frac{\partial v_4}{\partial v_3}\left(\frac{\partial v_5}{\partial v_4}\overline{v}_5\right) + \frac{\partial v_5}{\partial v_3}\overline{v}_5\right) \qquad \overline{v}_5 = \frac{\partial f}{\partial v_5} = 1$$

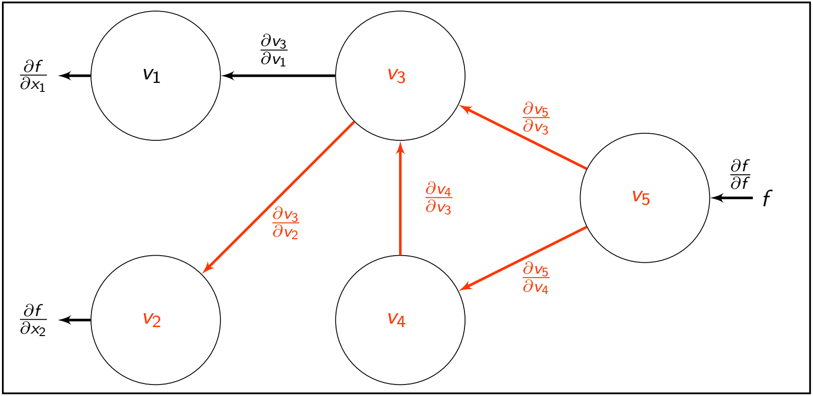
A shift in perspective



Apply chain rule to get gradient

$$\overline{v}_2 = \frac{\partial f}{\partial v_2} = \frac{\partial v_3}{\partial v_2}\left(\frac{\partial v_4}{\partial v_3}\left(\frac{\partial v_5}{\partial v_4}\overline{v}_5\right) + \frac{\partial v_5}{\partial v_3}\overline{v}_5\right)$$

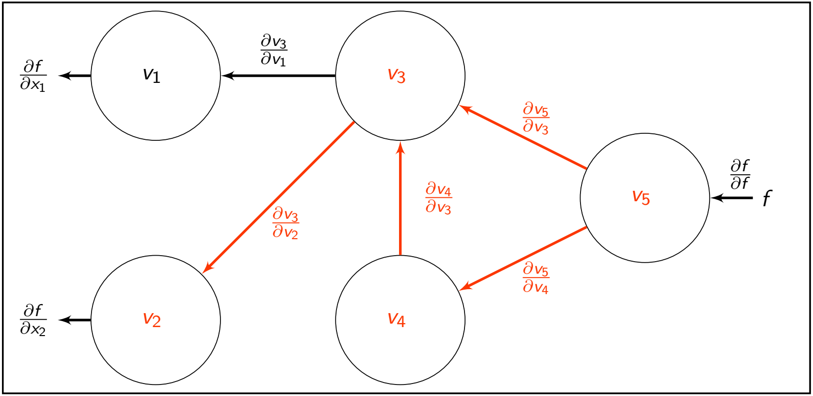
A shift in perspective



Apply chain rule to get gradient

$$\overline{v}_2 = \frac{\partial f}{\partial v_2} = \frac{\partial v_3}{\partial v_2}\left(\frac{\partial v_4}{\partial v_3}\overline{v}_4 + \frac{\partial v_5}{\partial v_3}\overline{v}_5\right)$$

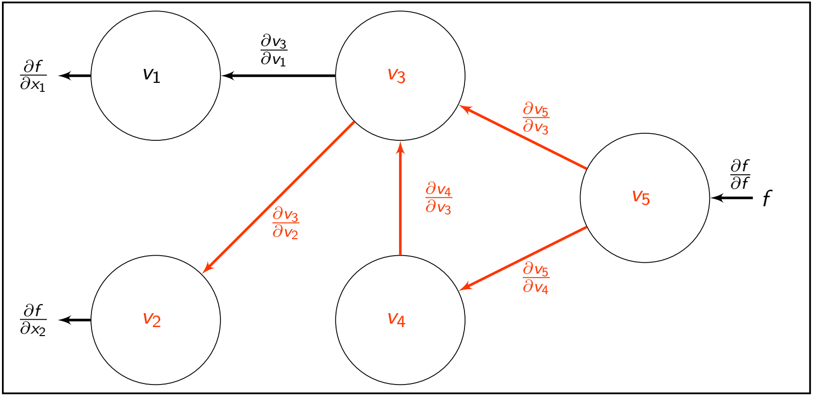
A shift in perspective



Apply chain rule to get gradient

$$\overline{v}_2 = \frac{\partial f}{\partial v_2} = \frac{\partial v_3}{\partial v_2}\overline{v}_3$$

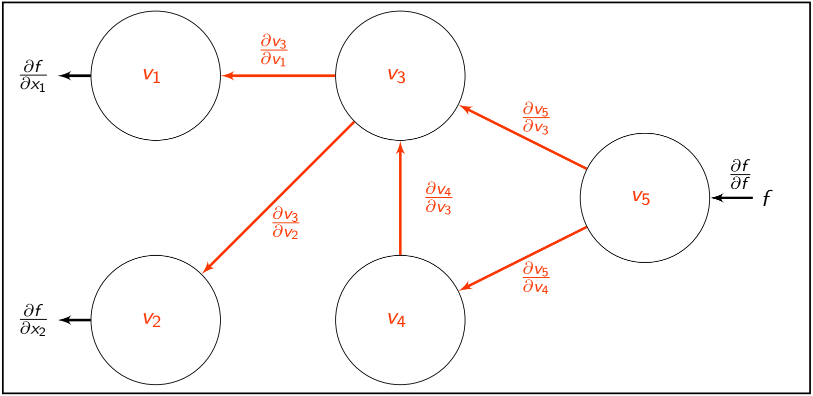
A shift in perspective



Apply chain rule to get gradient

$$\overline{v}_2 = \frac{\partial f}{\partial v_2}$$

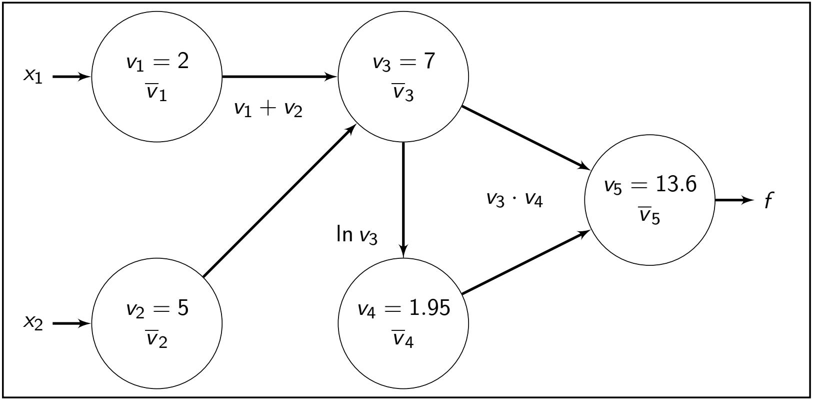
A shift in perspective



Apply chain rule to get gradient

$$\overline{v}_2 = \frac{\partial f}{\partial v_2} \qquad \overline{v}_1 = \frac{\partial f}{\partial v_1} = \frac{\partial v_3}{\partial v_1}\overline{v}_3$$

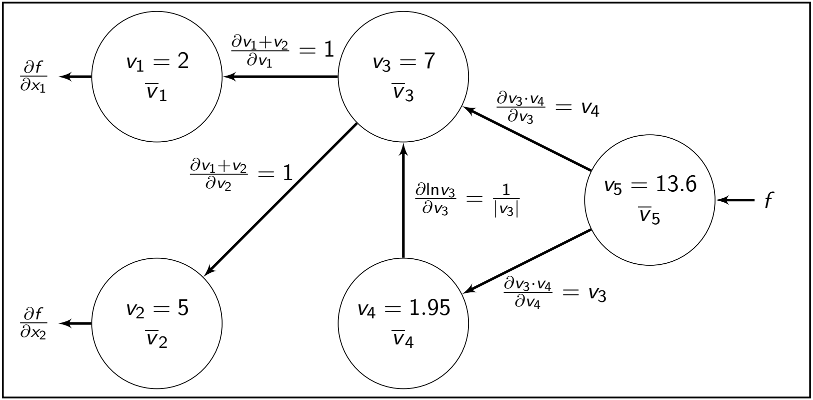
Reverse mode AD



$$f(x_1, x_2) = (x_1 + x_2) \cdot \ln(x_1 + x_2) \quad \rightsquigarrow \quad \nabla_{\mathbf{x}} f(2, 5) = \ ?$$

propagate $\overline{v}_k = \frac{\partial v_k}{\partial v_i} \overline{v}_i$ backwards through the graph

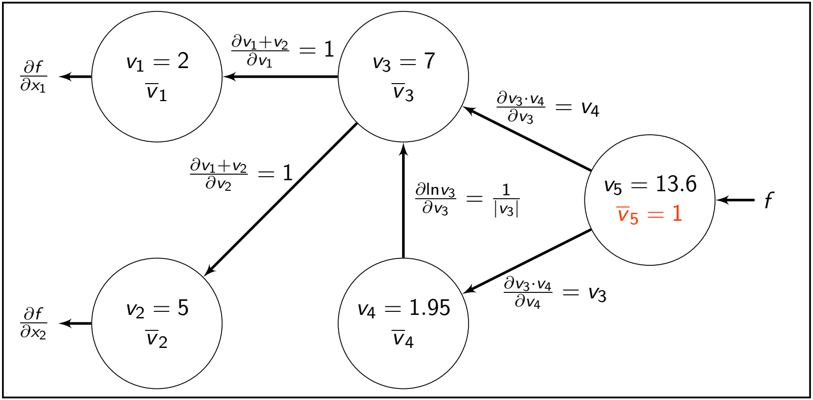
Reverse mode AD



$$f(x_1, x_2) = (x_1 + x_2) \cdot \ln(x_1 + x_2) \quad \rightsquigarrow \quad \nabla_{\mathbf{x}} f(2, 5) = \ ?$$

propagate $\overline{v}_k = \frac{\partial v_k}{\partial v_i} \overline{v}_i$ backwards through the graph

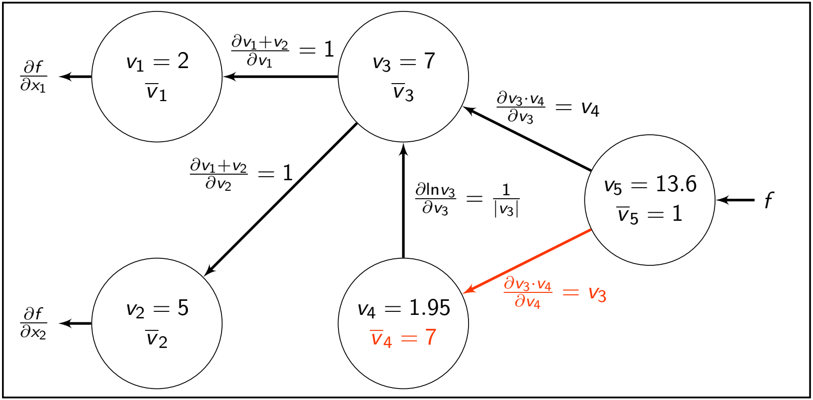
Reverse mode AD



$$f(x_1, x_2) = (x_1 + x_2) \cdot \ln(x_1 + x_2) \quad \rightsquigarrow \quad \nabla_{\mathbf{x}} f(2, 5) = \ ?$$

propagate $\overline{v}_k = \frac{\partial v_k}{\partial v_i} \overline{v}_i$ backwards through the graph

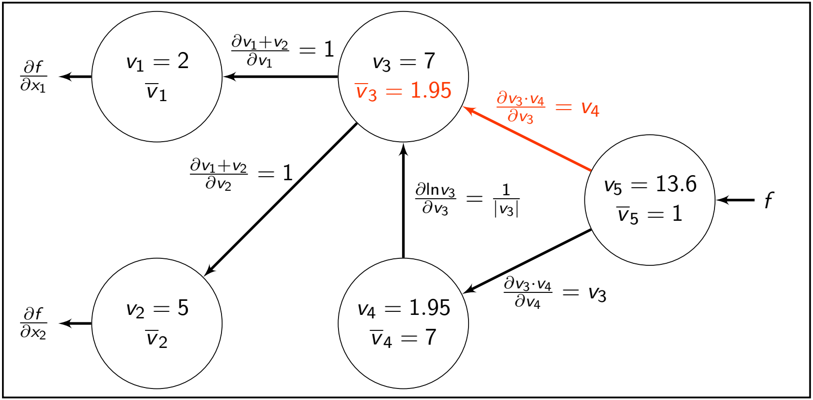
Reverse mode AD



$$f(x_1, x_2) = (x_1 + x_2) \cdot \ln(x_1 + x_2) \quad \rightsquigarrow \quad \nabla_{\mathbf{x}} f(2, 5) = \ ?$$

propagate $\overline{v}_k = \frac{\partial v_k}{\partial v_i} \overline{v}_i$ backwards through the graph

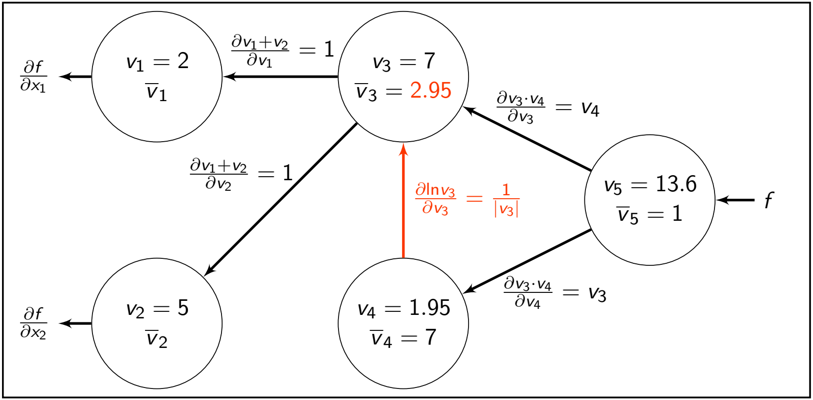
Reverse mode AD



$$f(x_1, x_2) = (x_1 + x_2) \cdot \ln(x_1 + x_2) \quad \rightsquigarrow \quad \nabla_{\mathbf{x}} f(2,5) = \ ?$$

propagate $\overline{v}_k = \frac{\partial v_k}{\partial v_i} \overline{v}_i$ backwards through the graph

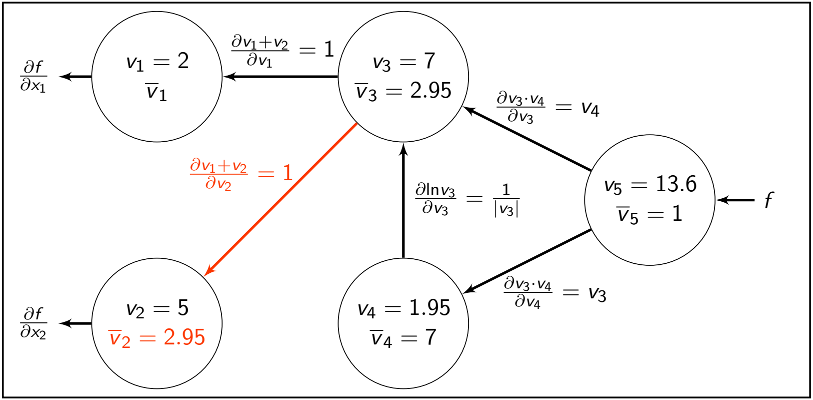
Reverse mode AD



$$f(x_1, x_2) = (x_1 + x_2) \cdot \ln(x_1 + x_2) \quad \rightsquigarrow \quad \nabla_{\mathbf{x}} f(2, 5) = \ ?$$

propagate $\overline{v}_k = \frac{\partial v_k}{\partial v_i} \overline{v}_i$ backwards through the graph

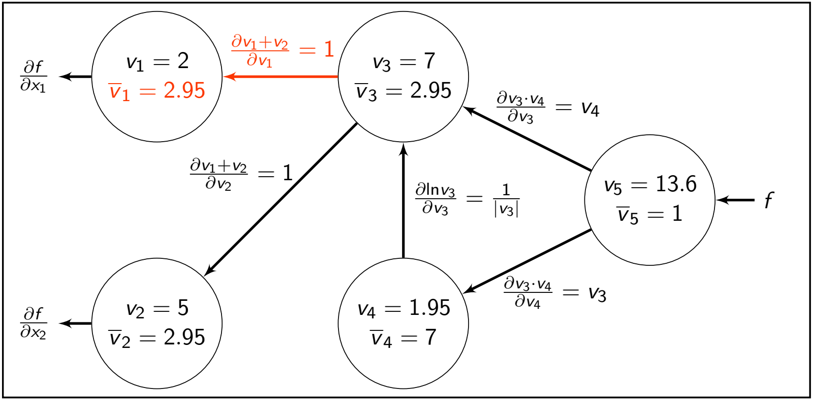
Reverse mode AD



$$f(x_1, x_2) = (x_1 + x_2) \cdot \ln(x_1 + x_2) \quad \rightsquigarrow \quad \nabla_{\mathbf{x}} f(2, 5) = \ ?$$

propagate $\overline{v}_k = \frac{\partial v_k}{\partial v_i} \overline{v}_i$ backwards through the graph

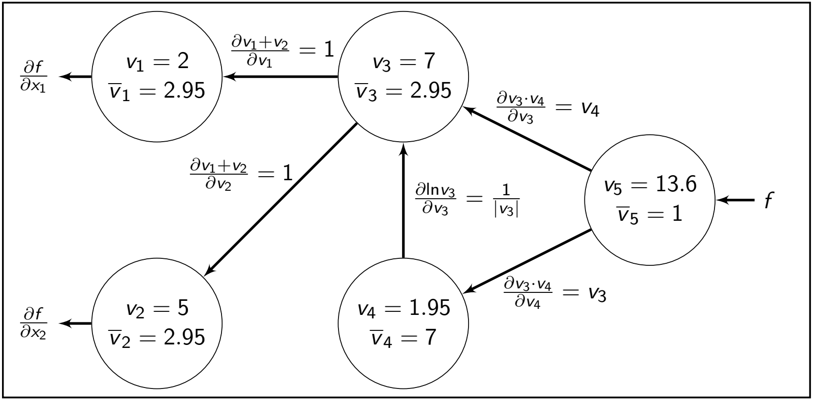
Reverse mode AD



$$f(x_1, x_2) = (x_1 + x_2) \cdot \ln(x_1 + x_2) \quad \rightsquigarrow \quad \nabla_{\mathbf{x}} f(2, 5) = \ ?$$

propagate $\overline{v}_k = \frac{\partial v_k}{\partial v_i} \overline{v}_i$ backwards through the graph

Reverse mode AD

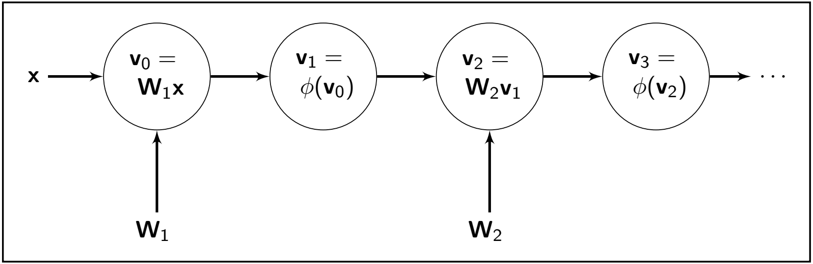


$$f(x_1, x_2) = (x_1 + x_2) \cdot \ln(x_1 + x_2) \quad \rightsquigarrow \quad \nabla_{\mathbf{x}} f(2,5) = \begin{pmatrix} 2.95 \\ 2.95 \end{pmatrix}$$

computes $\nabla_{\mathbf{x}} f(\mathbf{x}_0)$ exactly in one go (for each output)

Backpropagation does reverse mode AD!

Computation graph of the training error E

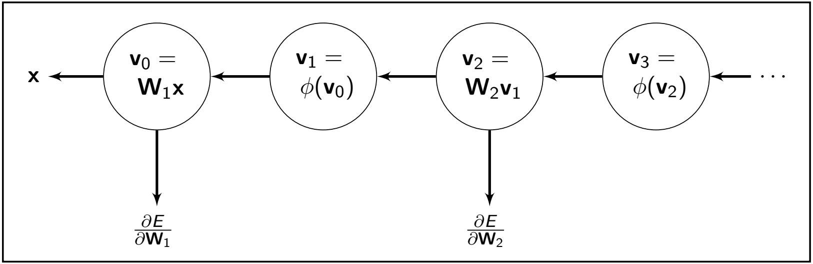


$$E(\mathbf{W}, \mathbf{x}, \mathbf{y}) = \|h(\mathbf{x}; \mathbf{W}) - \mathbf{y}\|^2$$

$$h(\mathbf{x}; \mathbf{W}) = \phi(\mathbf{W}_d \cdots \phi(\mathbf{W}_2 \phi(\mathbf{W}_1 \mathbf{x})) \cdots)$$

Backpropagation does reverse mode AD!

Computation graph of the training error E



$$E(\mathbf{W}, \mathbf{x}, \mathbf{y}) = \|h(\mathbf{x}; \mathbf{W}) - \mathbf{y}\|^2$$

$$h(\mathbf{x}; \mathbf{W}) = \phi(\mathbf{W}_d \cdots \phi(\mathbf{W}_2\phi(\mathbf{W}_1\mathbf{x}))\cdots)$$

Why bother?

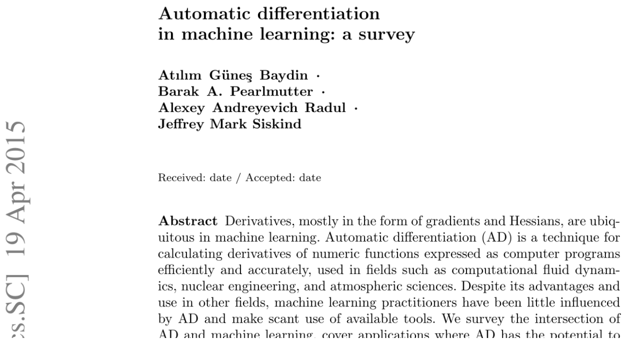
	scalable	efficient	exact	arbitrary code
Manual	✗	✓	✓	✗
Numerics	✓	✗	✗	✓
Symbolic	(✓)	(✓)	✓	✗
Autodiff	✓	(✓)	✓	✓

. . . plenty of libraries available at `autodiff.org`

Demo!

Recommended reading

A. G. Baydin, B. A. Pearlmutter, A. A. Radul, J. M. Siskind: **Automatic differentiation in machine learning: a survey**, *arXiv preprint arXiv: 1502.05767v2*, 2015



Automatic differentiation in machine learning: a survey

Atılım Güneş Baydin ·
Barak A. Pearlmutter ·
Alexey Andreyevich Radul ·
Jeffrey Mark Siskind

Received: date / Accepted: date

Abstract Derivatives, mostly in the form of gradients and Hessians, are ubiquitous in machine learning. Automatic differentiation (AD) is a technique for calculating derivatives of numeric functions expressed as computer programs efficiently and accurately, used in fields such as computational fluid dynamics, nuclear engineering, and atmospheric sciences. Despite its advantages and use in other fields, machine learning practitioners have been little influenced by AD and make scant use of available tools. We survey the intersection of AD and machine learning, cover applications where AD has the potential to