How to compute Derivatives

derivatives appear very often in Computer Science, under several forms

$$f : \Re \to \Re \qquad f' \equiv \frac{df}{dx},\ f'' \equiv \frac{d^2 f}{dx^2}, \cdots$$

$$f : \Re^n \to \Re \qquad \frac{\partial f}{\partial x}, \frac{\partial^2 f}{\partial x^2}, \cdots, \nabla f, \mathbf{H}_f, \cdots$$

$$f : \Re^n \to \Re^m \qquad \mathbf{J}_f, \cdots$$

How to compute Derivatives

we have 3 approaches

1) **Numerical** Differentiation

2) **Symbolic** Differentiation

3) **Automatic** Differentiation

Symbolic Differentiation

compute the **symbolic expression** of the derivative of a function

it needs manipulation in computer algebra systems such as Mathematica, Maxima, Maple, Matlab’s Symbolic Toolbox, Python’s sympy

input:

- the symbolic expression of the function

output:

- the symbolic expression of the derivative

```
>> syms x
>> diff(sin(x))
ans =
  cos(x)

>> diff(sin(x),2)
ans =
  -sin(x)

>> diff(x*sin(2*x))
ans =
   sin(2*x) + 2*x*cos(2*x)
```

```
>> syms y
>> diff(x*y^2+cos(x*y),x)
ans =
  y^2 - y*sin(x*y)

>> diff(x*y^2+cos(x*y),y)
ans =
  2*x*y - x*sin(x*y)

>> diff(x*y^2+cos(x*y),x,2)
ans =
  -y^2*cos(x*y)
```

```
>> f(x,y) = x*y^2+cos(x*y);
>> gradient(f,[x,y])
ans(x, y) =
   y^2 - y*sin(x*y)
 2*x*y - x*sin(x*y)

>> laplacian(f,[x,y])
ans(x, y) =
 2*x - x^2*cos(x*y) - y^2*cos(x*y)

>> diff(f,x,2)+diff(f,y,2)
ans(x, y) =
 2*x - x^2*cos(x*y) - y^2*cos(x*y)
```

Automatic Differentiation (AD)

compute the exact value of derivatives of a function **at a given point** (called differentiation point)

its basic idea is to apply repeatedly the chain rule from Calculus

input:

- the point where the derivative has to be evaluated
- the **symbolic expression** of the function **OR** a **program** that evaluates the function at any point

Automatic Differentiation (AD)

compute the exact value of derivatives of a function **at a given point** (called differentiation point)

the idea is to apply repeatedly the chain rule from Calculus to either the basic functions that compose the function to be differentiated or the lines of the program that implements such function

output:

- the exact value of the derivative at the differentiation point

Automatic Differentiation (AD)

AD **does no**t compute either the symbolic expression of the derivatives or generate a program that implements such an expression

- AD, like finite difference, computes the value of the derivatives **at a given point**
- unlike finite difference, AD provides the **exact** value of the derivatives
- if we want to evaluate the derivative (gradient,..) of the same function at several distinct points we need to run **several times** the AD code

Automatic Differentiation (AD)

- AD is used in many Machine Learning optimization problems, e.g. deep neural networks, which are parametrized by (millions/billions of) weights that are computed by **Stochastic Gradient descent** or its variants (GD with momentum, ADAM,..) to find the minimum of the loss function defined in terms of a trained set
- AD may also be used in Newton's minimization method and Levenberg-Marquardt method

Automatic Differentiation (AD)

AD may operate in two ways:

- ✓ **Forward** mode
- ✓ **Reverse** mode (also called **Backpropagation**)

Forward mode evaluates a derivative of a function at a given point by performing elementary derivative operations concurrently with the operations of evaluating the function itself at a given point

AD relies on the **computational graph** that represents the function to be differentiated

Automatic Differentiation (AD)

AD may operate in two ways:

✓ **Forward** mode

✓ **Reverse** mode (also called **Backpropagation**)

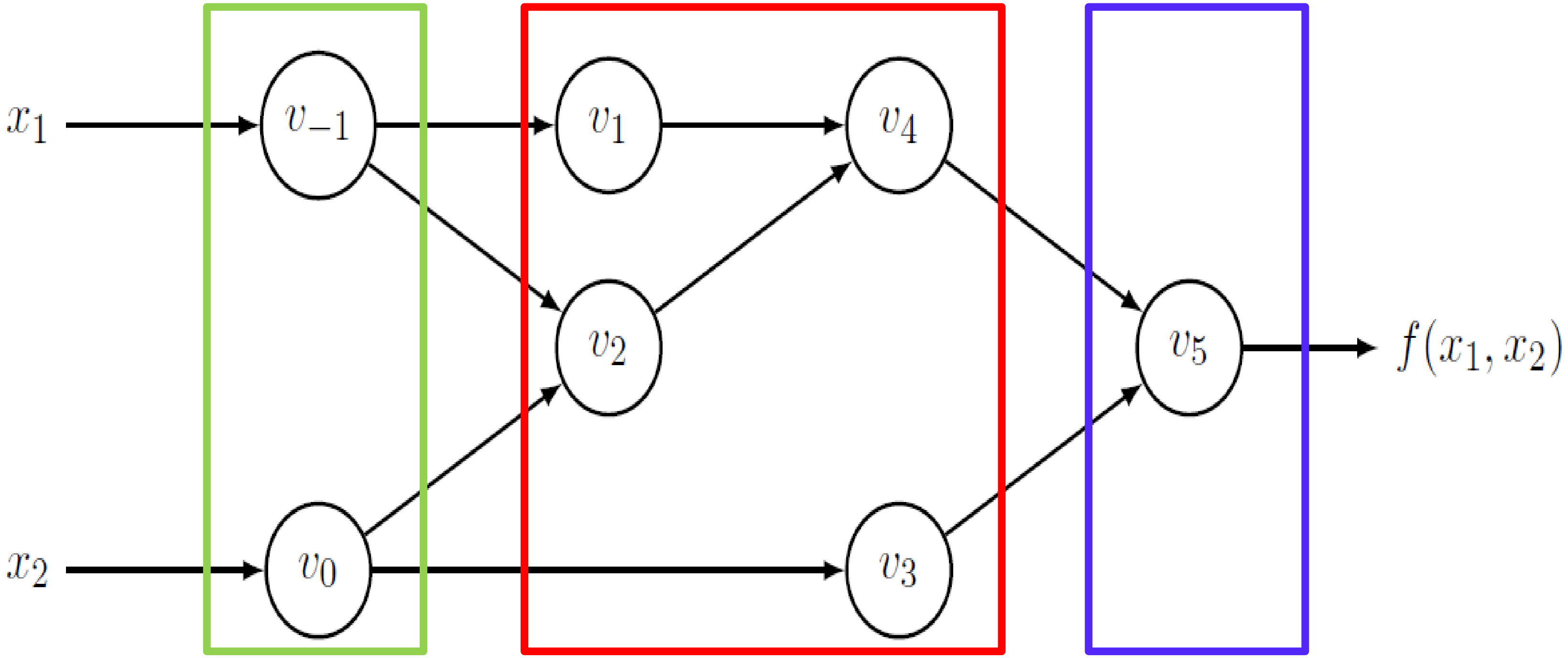
Reverse mode uses an extension of the forward mode computational graph to enable the computation of derivatives (gradient) by a reverse traversal of the computational graph

AD relies on the **computational graph** that represents the function to be differentiated

Automatic Differentiation (AD)

Forward mode
evaluation (differentiation) point: (2,5)

$$f(x_1, x_2) = log(x_1) + x_1 x_2 - sin(x_2)$$

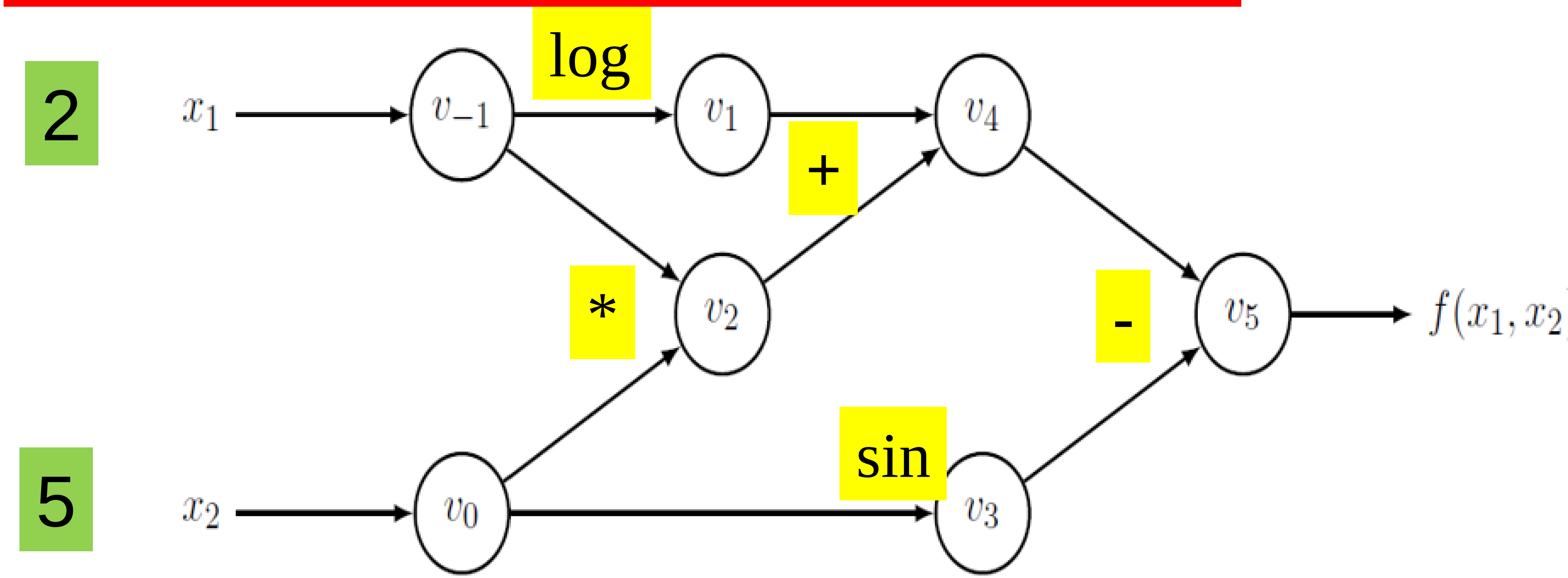


computational graph of f

Automatic Differentiation (AD)

Forward mode

evaluation (differentiation) point: (2,5)



a forward traversal of the computational graph (i.e. from left to right) allows us to compute $f(2,5)$

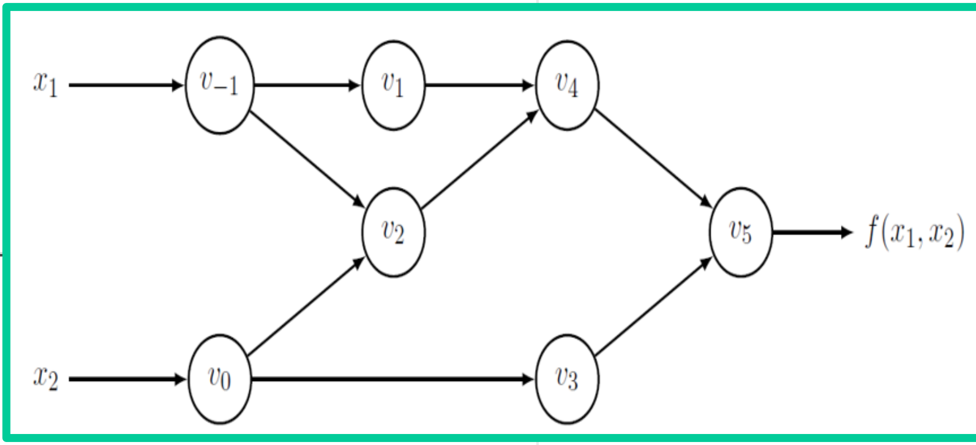
Automatic Differentiation (AD)

Forward mode

$$f(x_1, x_2) = log(x_1) + x_1 x_2 - sin(x_2)$$

Forward Primal Trace

$$\begin{aligned}
v_{-1} &= x_1 && = 2 \\
v_0 &= x_2 && = 5 \\
\hline
v_1 &= \ln v_{-1} && = \ln 2 \\
v_2 &= v_{-1} \times v_0 && = 2 \times 5 \\
v_3 &= \sin v_0 && = \sin 5 \\
v_4 &= v_1 + v_2 && = 0.693 + 10 \\
v_5 &= v_4 - v_3 && = 10.693 + 0.959 \\
\hline
y &= v_5 && = 11.652
\end{aligned}$$



Automatic Differentiation (AD)

Forward mode

$$f(x_1, x_2) = log(x_1) + x_1 x_2 - sin(x_2)$$

using the computational graph enriched with the derivatives of the variables ($\dot{v}_i$) we compute the value of the gradient, by first computing the value of the partial derivative respect to x_1 and then the value of the partial derivative respect to x_2

Automatic Differentiation (AD)

Forward mode

$$f(x_1, x_2) = log(x_1) + x_1 x_2 - sin(x_2)$$

derivatives are computed by recursively applying the **chain rule**

the **chain rule** is the formula that expresses the derivative of the composition of two differentiable functions in terms of the derivatives of each function recap **chain rule**

remind the chain rule in one-dimension:
given

$$z = f(y) = f(g(x))$$

$$y = g(x)$$

then

$$\frac{df}{dx} = \frac{df}{dy}\frac{dy}{dx}$$

recap **chain rule**

remind the chain rule in two-dimension:
given

$$z = f(y_1, y_2) = f(y_1(x_1, x_2), y_2(x_1, x_2))$$

then

$$\frac{\partial f}{\partial x_1} = \frac{\partial f}{\partial y_1}\frac{\partial y_1}{\partial x_1} + \frac{\partial f}{\partial y_2}\frac{\partial y_2}{\partial x_1}$$

$$\frac{\partial f}{\partial x_2} = \frac{\partial f}{\partial y_1}\frac{\partial y_1}{\partial x_2} + \frac{\partial f}{\partial y_2}\frac{\partial y_2}{\partial x_2}$$

Automatic Differentiation (AD)

Forward mode

$$f(x_1, x_2) = log(x_1) + x_1 x_2 - sin(x_2)$$

Forward Primal Trace

$$\begin{aligned}
v_{-1} &= x_1 && = 2 \\
v_0 &= x_2 && = 5 \\
\hline
v_1 &= \ln v_{-1} && = \ln 2 \\
v_2 &= v_{-1} \times v_0 && = 2 \times 5 \\
v_3 &= \sin v_0 && = \sin 5 \\
v_4 &= v_1 + v_2 && = 0.693 + 10 \\
v_5 &= v_4 - v_3 && = 10.693 + 0.959 \\
\hline
y &= v_5 && = 11.652
\end{aligned}$$

Forward Tangent (Derivative) Trace

$$\frac{\partial f}{\partial x_1}$$

$$\begin{aligned}
\dot{v}_{-1} &= \dot{x}_1 && = 1 \\
\dot{v}_0 &= \dot{x}_2 && = 0 \\
\hline
\dot{v}_1 &= \dot{v}_{-1}/v_{-1} && = 1/2 \\
\dot{v}_2 &= \dot{v}_{-1} \times v_0 + \dot{v}_0 \times v_{-1} && = 1 \times 5 + 0 \times 2 \\
\dot{v}_3 &= \dot{v}_0 \times \cos v_0 && = 0 \times \cos 5 \\
\dot{v}_4 &= \dot{v}_1 + \dot{v}_2 && = 0.5 + 5 \\
\dot{v}_5 &= \dot{v}_4 - \dot{v}_3 && = 5.5 - 0 \\
\hline
\dot{y} &= \dot{v}_5 && \frac{\partial f}{\partial x_1} = \mathbf{5.5}
\end{aligned}$$

Automatic Differentiation (AD)

Forward mode

$$f(x_1, x_2) = log(x_1) + x_1 x_2 - sin(x_2)$$

the choice $\dot{x_1} = 1, \dot{x_2} = 0$

implies that we are computing $\dfrac{\partial f}{\partial x_1}$

the choice $\dot{x_1} = 0, \dot{x_2} = 1$

implies that we are computing $\dfrac{\partial f}{\partial x_2}$

Automatic Differentiation (AD)

Forward mode

$$f(x_1, x_2) = log(x_1) + x_1 x_2 - sin(x_2)$$

$$\frac{\partial f}{\partial x_2}$$

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	$= v_5$	$= 11.652$

$\dot{v}_{-1} = \dot{x}_1 = 0 \qquad \dot{v}_0 = \dot{x}_2 = 1$

$$\dot{v}_1 = \dot{v}_{-1} / v_{-1} = 0/2 = 0$$

$$\dot{v}_2 = \dot{v}_{-1} \times v_0 + \dot{v}_0 \times v_{-1} = 0 \times 5 + 1 \times 2 = 2$$

$$\dot{v}_3 = \dot{v}_0 \times cos(v_0) = 1 \times cos(5)$$

$$\dot{v}_4 = \dot{v}_1 + \dot{v}_2 = 0 + 2$$

$$\dot{v}_5 = \dot{v}_4 - \dot{v}_3 = 2 - cos(5) = 1.7163$$

$$\dot{v}_5 = 1.7163 \leftarrow \frac{\partial f}{\partial x_2}$$

Automatic Differentiation (AD)

Forward mode

$$f(x_1, x_2) = log(x_1) + x_1 x_2 - sin(x_2)$$

we need **2** (forward) traversals of the tangent computational graph to compute the gradient of f,
one for each component of the gradient

with n variables, we need $\boldsymbol{n}$ (forward) traversals to compute the gradient of f

Automatic Differentiation (AD)

Reverse mode

$$f(x_1, x_2) = log(x_1) + x_1 x_2 - sin(x_2)$$

the Reverse mode corresponds to a generalized **backpropagation** algorithm, in that it propagates derivatives backward from the output y toward the input variables x_1 and x_2.

Reverse mode must be used **after** the forward traversal of the computational graph

Automatic Differentiation (AD)

Reverse mode

$$f(x_1, x_2) = log(x_1) + x_1 x_2 - sin(x_2)$$

Reverse mode requires to complement each **intermediate variable** with the so called **adjoint variable**

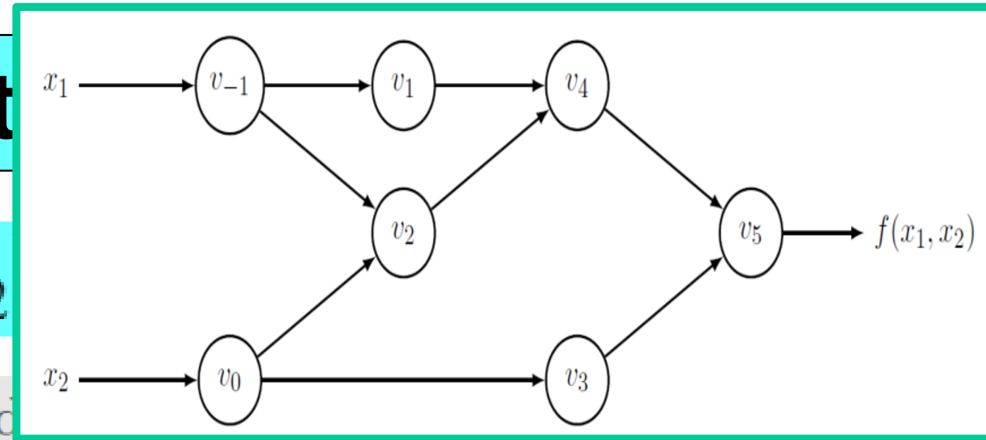
$$\bar{v}_i = \frac{\partial y_j}{\partial v_i}$$

the adjoint variable $\bar{v}_i$ represents the sensitivity of a considered output with respect to changes in v_i

Automatic Different

Reverse mode

$f(x_1, x_2$



Forward Primal Trace

$$
\begin{aligned}
v_{-1} &= x_1 && = 2 \\
v_0 &= x_2 && = 5 \\
\hline
v_1 &= \ln v_{-1} && = \ln 2 \\
v_2 &= v_{-1} \times v_0 && = 2 \times 5 \\
v_3 &= \sin v_0 && = \sin 5 \\
v_4 &= v_1 + v_2 && = 0.693 + 10 \\
v_5 &= v_4 - v_3 && = 10.693 + 0.959 \\
\hline
y &= v_5 && = 11.652
\end{aligned}
$$

Reverse Ad

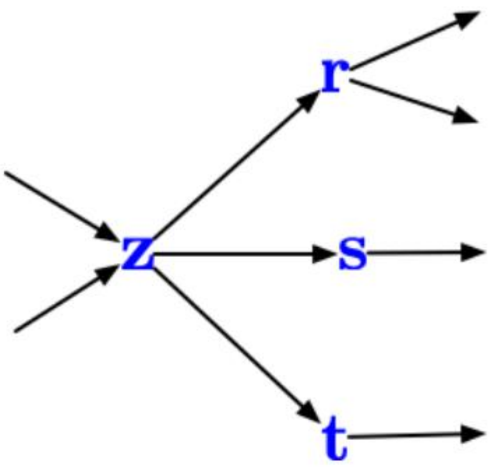
$$
\begin{aligned}
\bar{x}_1 &= \bar{v}_{-1} && && = 5.5 \\
\bar{x}_2 &= \bar{v}_0 && && = 1.716 \\
\hline
\bar{v}_{-1} &= \bar{v}_{-1} + \bar{v}_1 \frac{\partial v_1}{\partial v_{-1}} && = \bar{v}_{-1} + \bar{v}_1 / v_{-1} && = 5.5 \\
\bar{v}_0 &= \bar{v}_0 + \bar{v}_2 \frac{\partial v_2}{\partial v_0} && = \bar{v}_0 + \bar{v}_2 \times v_{-1} && = 1.716 \\
\bar{v}_{-1} &= \bar{v}_2 \frac{\partial v_2}{\partial v_{-1}} && = \bar{v}_2 \times v_0 && = 5 \\
\bar{v}_0 &= \bar{v}_3 \frac{\partial v_3}{\partial v_0} && = \bar{v}_3 \times \cos v_0 && = -0.284 \\
\bar{v}_2 &= \bar{v}_4 \frac{\partial v_4}{\partial v_2} && = \bar{v}_4 \times 1 && = 1 \\
\bar{v}_1 &= \bar{v}_4 \frac{\partial v_4}{\partial v_1} && = \bar{v}_4 \times 1 && = 1 \\
\bar{v}_3 &= \bar{v}_5 \frac{\partial v_5}{\partial v_3} && = \bar{v}_5 \times (-1) && = -1 \\
\bar{v}_4 &= \bar{v}_5 \frac{\partial v_5}{\partial v_4} && = \bar{v}_5 \times 1 && = 1 \\
\hline
\bar{v}_5 &= \bar{y} && = 1
\end{aligned}
$$

Automatic Differentiation (AD)

Reverse mode

the general rule for $\bar{v}_i$ is

$$\bar{v}_i = \sum_{v_j children(v_i)} \bar{v}_j \frac{\partial v_j}{\partial v_i}$$

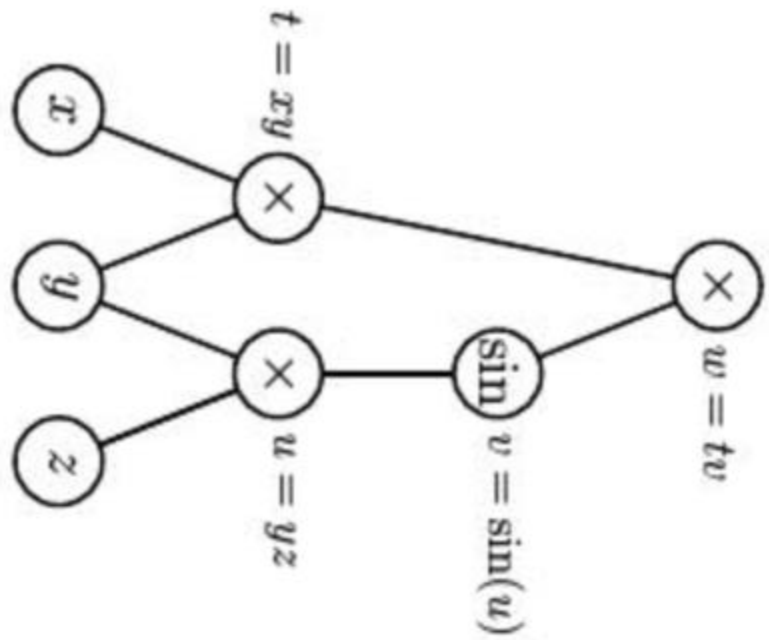


$$\bar{z} = \bar{r}\frac{\partial r}{\partial z} + \bar{s}\frac{\partial s}{\partial z} + \bar{t}\frac{\partial t}{\partial z}.$$

Automatic Differentiation (AD)

one more example $$f\left(x, y, z\right) = xy\sin(yz)$$

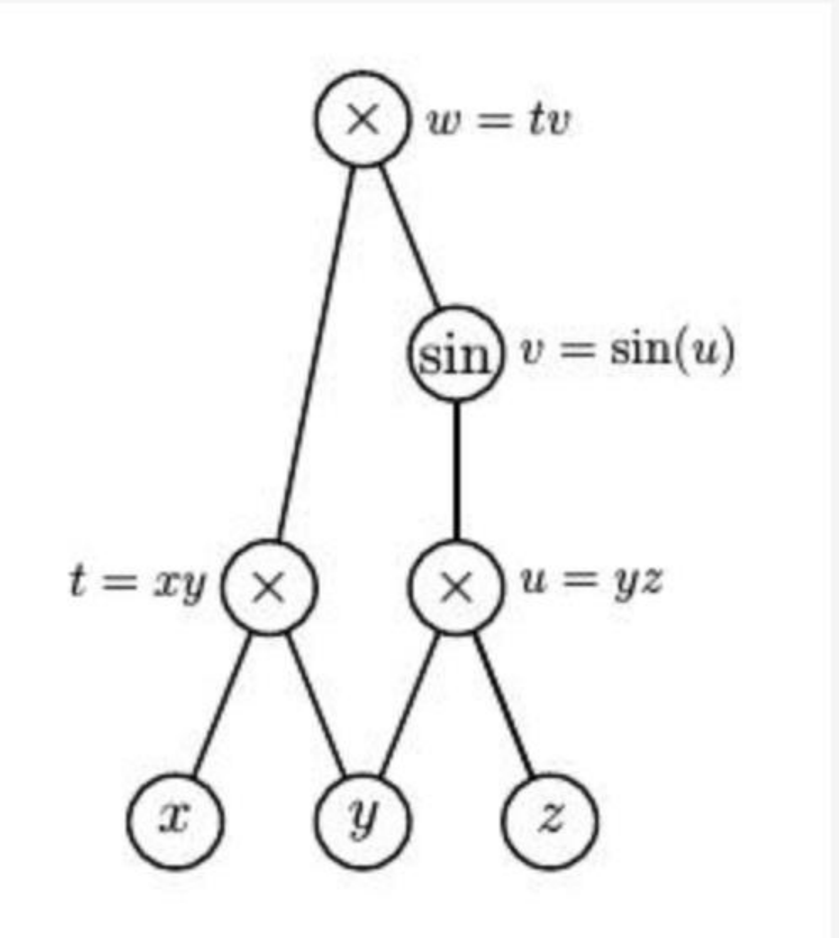
computational graph of f



Automatic Differentiation (AD)

one more example $f(x, y, z) = xy\sin(yz)$

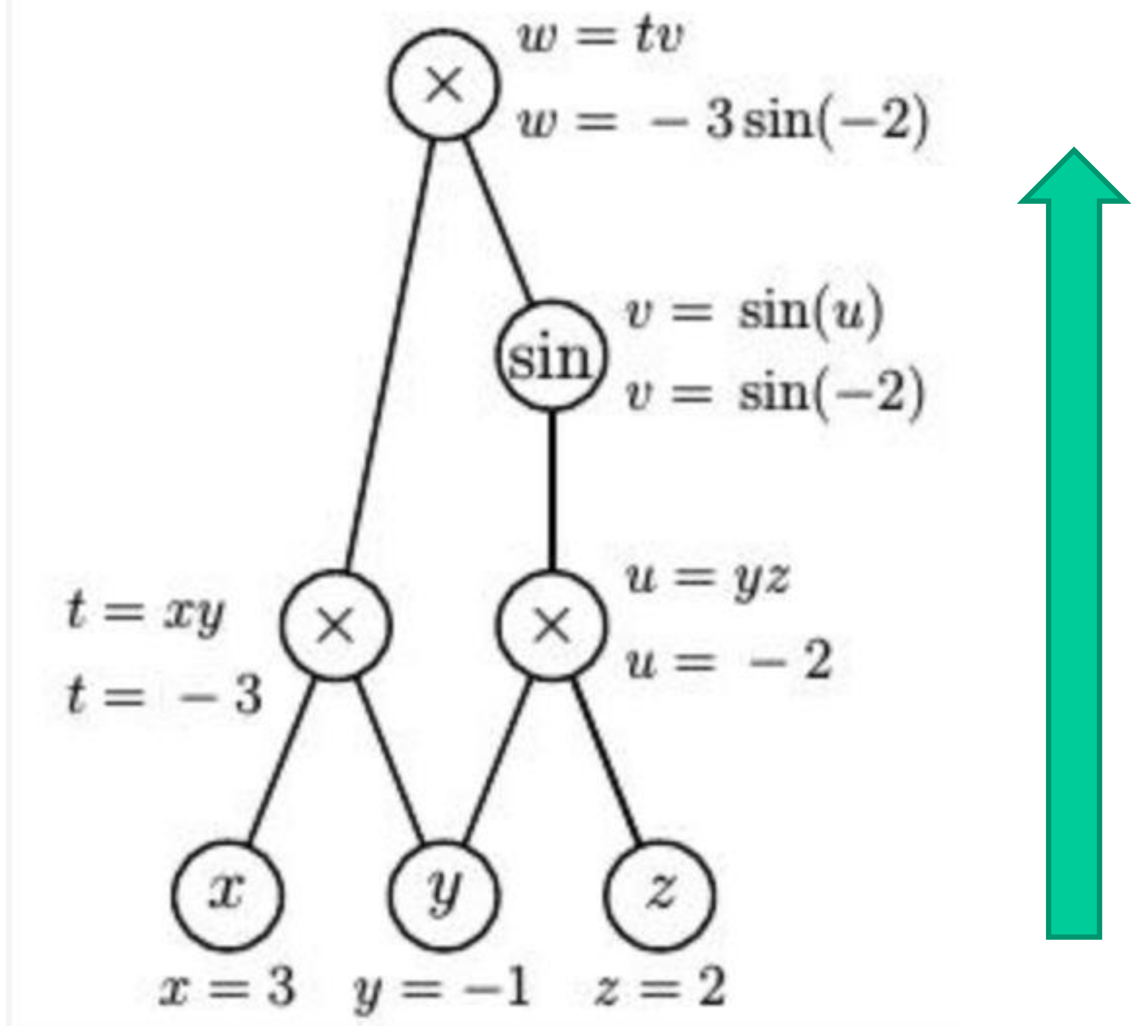
computational graph of f



Automatic Differentiation (AD)

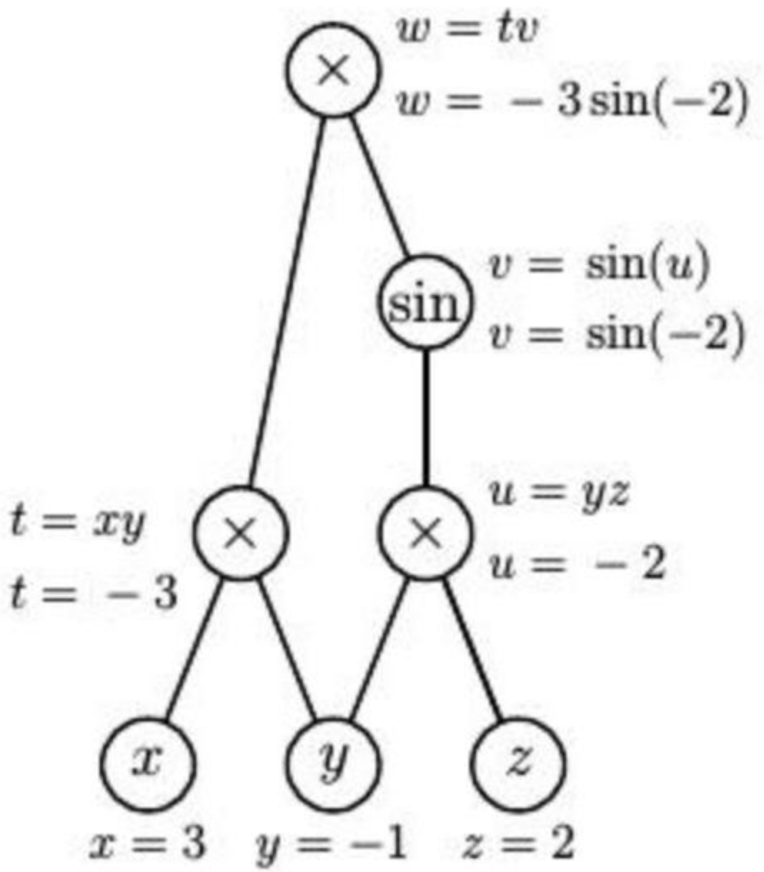
Forward mode

Forward pass on the computational graph (evaluating *f* at the differentiation point (3,-1,2)



Reverse mode

Backward pass on
the computational graph



$$\bar{w} = \frac{\partial w}{\partial w} = 1.$$

$$\bar{v} = \frac{\partial w}{\partial v} = t = -3$$

$$\bar{t} = \frac{\partial w}{\partial t} = v = \sin(-2)$$

$$\bar{u} = \frac{\partial w}{\partial v}\frac{\partial v}{\partial u} = \bar{v}\cos(u) = -3\cos(-2)$$

$$\bar{x} = \frac{\partial w}{\partial t}\frac{\partial t}{\partial x} = \bar{t}\,y = -\sin(-2)$$

$$\begin{aligned}\bar{y} &= \frac{\partial w}{\partial t}\frac{\partial t}{\partial y} + \frac{\partial w}{\partial u}\frac{\partial u}{\partial y} \\ &= \bar{t}\,x + \bar{u}z \\ &= 3\sin(-2) - 6\cos(-2)\end{aligned}$$

$$\bar{z} = \frac{\partial w}{\partial u}\frac{\partial u}{\partial z} = \bar{u}y = 3\cos(-2).$$

Automatic Differentiation (AD)

Reverse mode

note that the Reverse mode needs just **one forward pass** and **one reverse pass** to compute the gradient (all partial derivatives) at a given point, regardless of the number of independent variables

Reverse mode is **more efficient** than Forward mode in computing gradient of functions of several variables

Automatic Differentiation (AD) in Matlab

```
>> x0=2; y0=5;
>> xdl = dlarray([x0,y0]);
>>[fval,AD_grad]=dlfeval(@simplefg1,xdl)
fval =
  1×1 dlarray
   11.6521
AD_grad =
  1×2 dlarray
    5.5000    1.7163
```

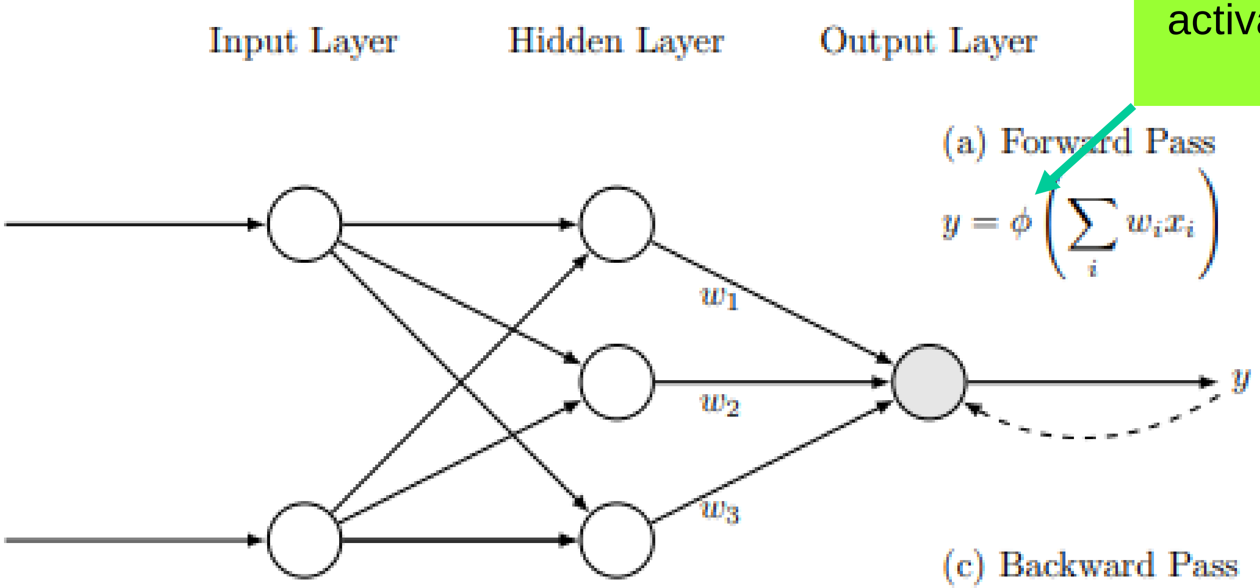
```
function [f,grad] = simplefg1(x)
f = log(x(1))+x(1)*x(2)-sin(x(2));
grad = dlgradient(f,x);
end
```

AD in Machine Learning

training of a (deep) neural network $\Longrightarrow$ find weights w_i

the computational graph is the neural network

activation function (i.e. ReLU)



(a) Forward Pass

$$y = \phi\left(\sum_i w_i x_i\right)$$

(b) Error at the Output

$$E(w_1, w_2, \ldots, w_m) = \left\|\mathbf{t} - \mathbf{y}(w_1, w_2, \ldots, w_m)\right\|_2^2$$

Loss function

(c) Backward Pass

$$\Delta w_i = -\eta \frac{\partial E}{\partial w_i}$$

$$\boxed{\frac{\partial E}{\partial w_i} = \bar{w}_i}$$

training sample

x_1, x_2, t

$$\min_{w_1, w_2, \ldots, w_m} E(w_1, w_2, \ldots, w_m)$$

Gradient Descent, learning rate η

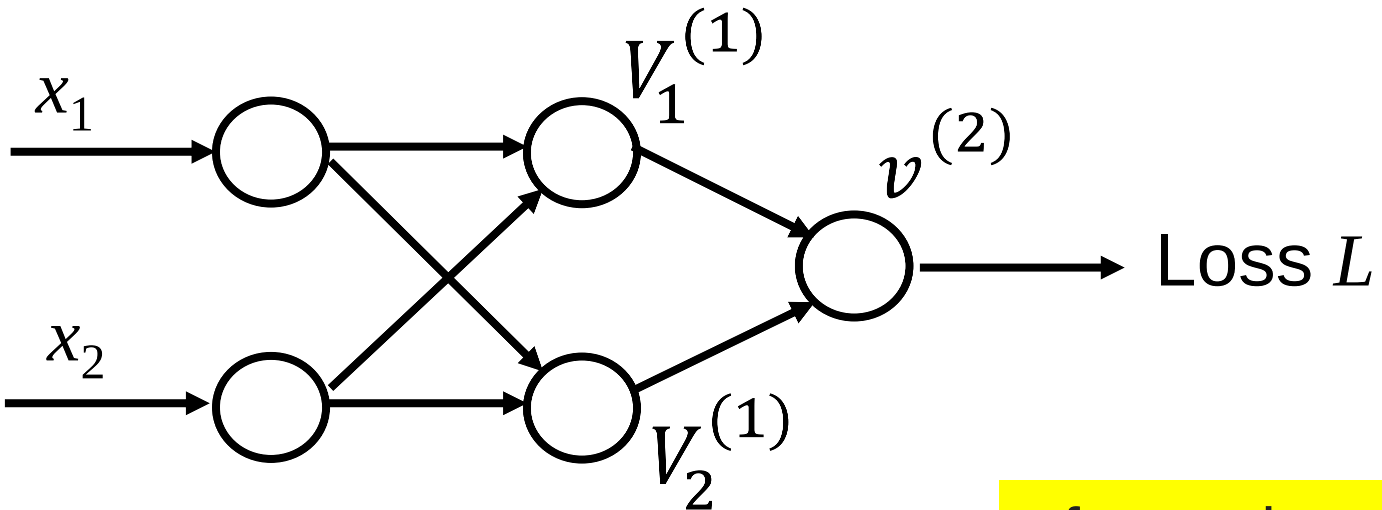
update w_i

$$\mathbf{w}^{(k+1)} = \mathbf{w}^{(k)} - \eta \nabla E\left(\mathbf{w}^{(k)}\right)$$

AD in Machine Learning

training of a neural network

Neural network with two linear layers



forward mode

$$V_1^{(1)} = W_{11}^{(1)} x_1 + W_{12}^{(1)} x_2 + b_1^{(1)}$$

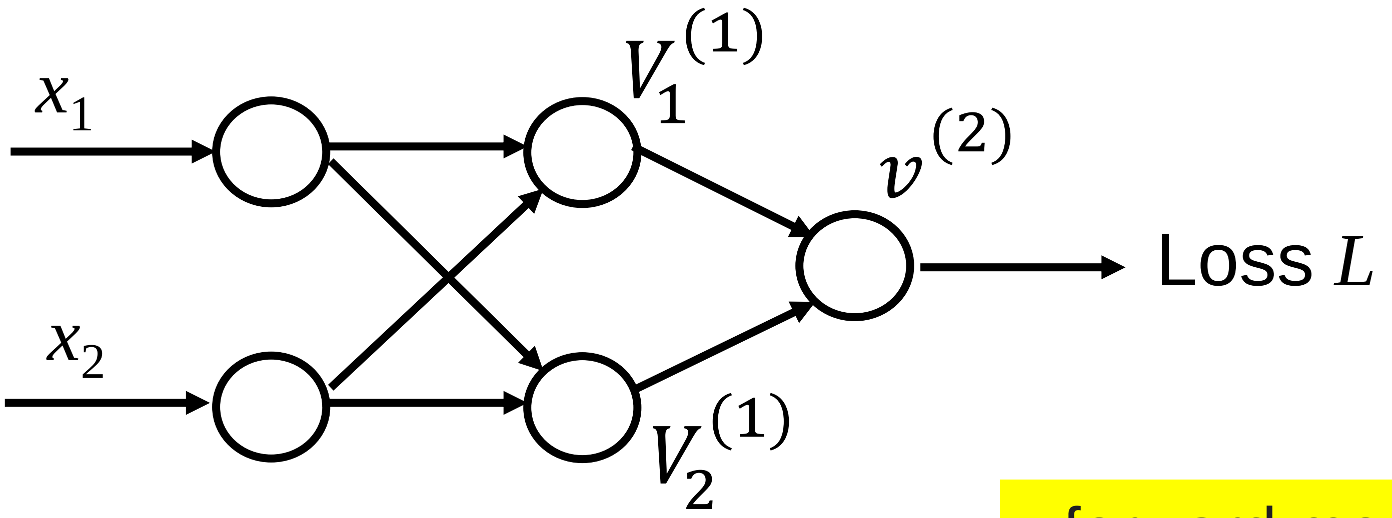
$$V_2^{(1)} = W_{21}^{(1)} x_1 + W_{22}^{(1)} x_2 + b_2^{(1)}$$

$$v^{(2)} = w_1^{(2)} V_1^{(1)} + w_2^{(2)} V_2^{(1)} + b^{(2)}$$

AD in Machine Learning

training of a neural network

Neural network with two linear layers



forward mode

$$V^{(1)} = W^{(1)} x + b^{(1)}$$

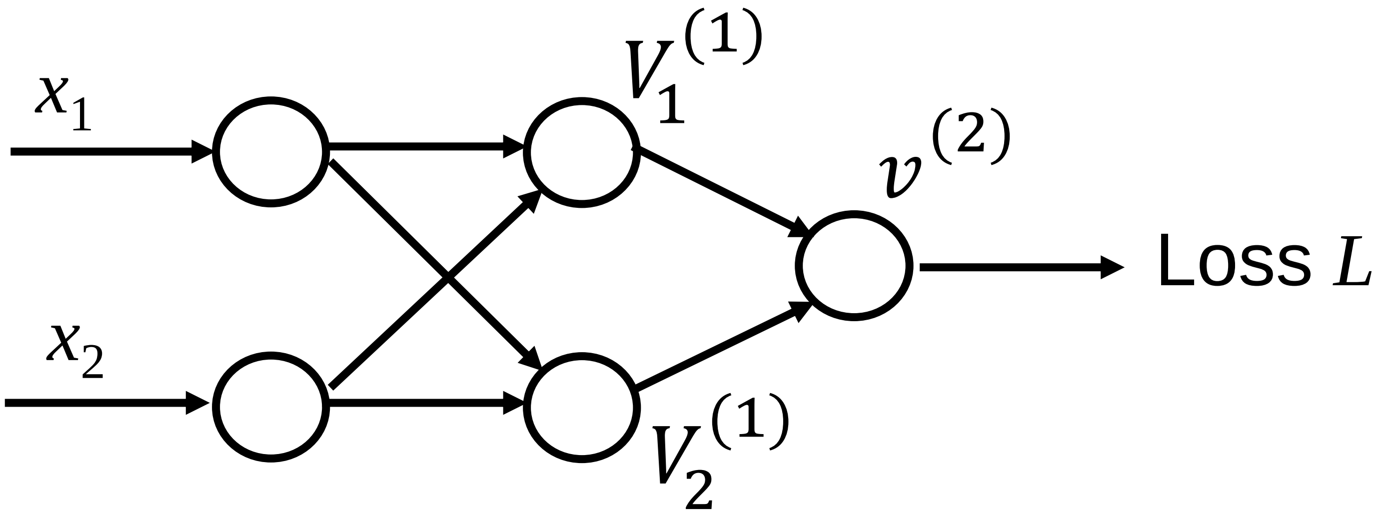
$$v^{(2)} = w^{(2)T} V^{(1)} + b^{(2)}$$

matrix-vector notation

AD in Machine Learning

training of a neural network

Neural network with two linear layers



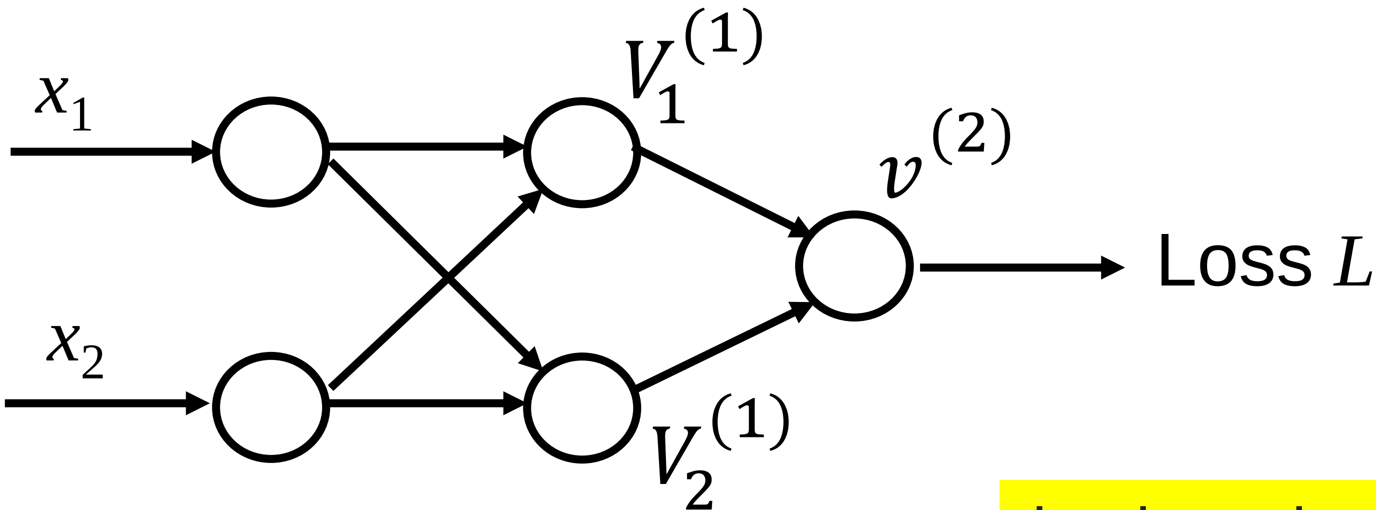
$$L = \frac{1}{|T|} \sum_{|T|} \left(v^{(2)} - T_{true}\right)^2$$

Training set T

AD in Machine Learning

training of a neural network

Neural network with two linear layers



backward mode

$$L = \frac{1}{|T|} \sum_{|T|} \left(v^{(2)} - T_{true}\right)^2$$

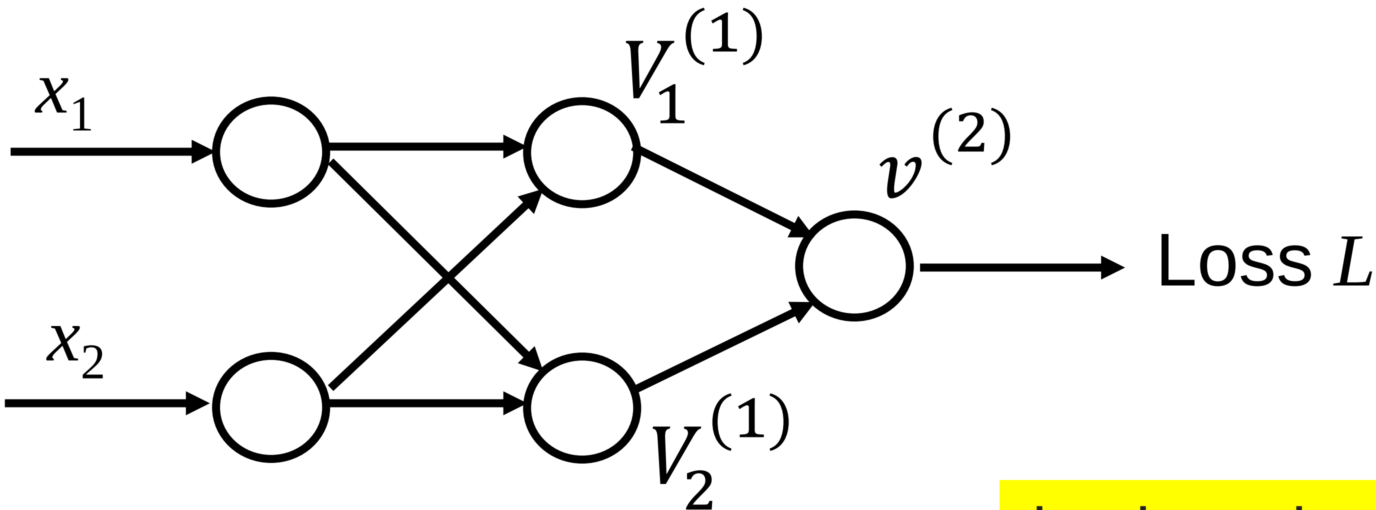
$$\frac{\partial L}{\partial v^{(2)}} = \frac{2}{|T|} \left(v^{(2)} - T_{true}\right)$$

this is the gradient of the Loss respect to each individual prediction of the network

AD in Machine Learning

training of a neural network

Neural network with two linear layers



backward mode

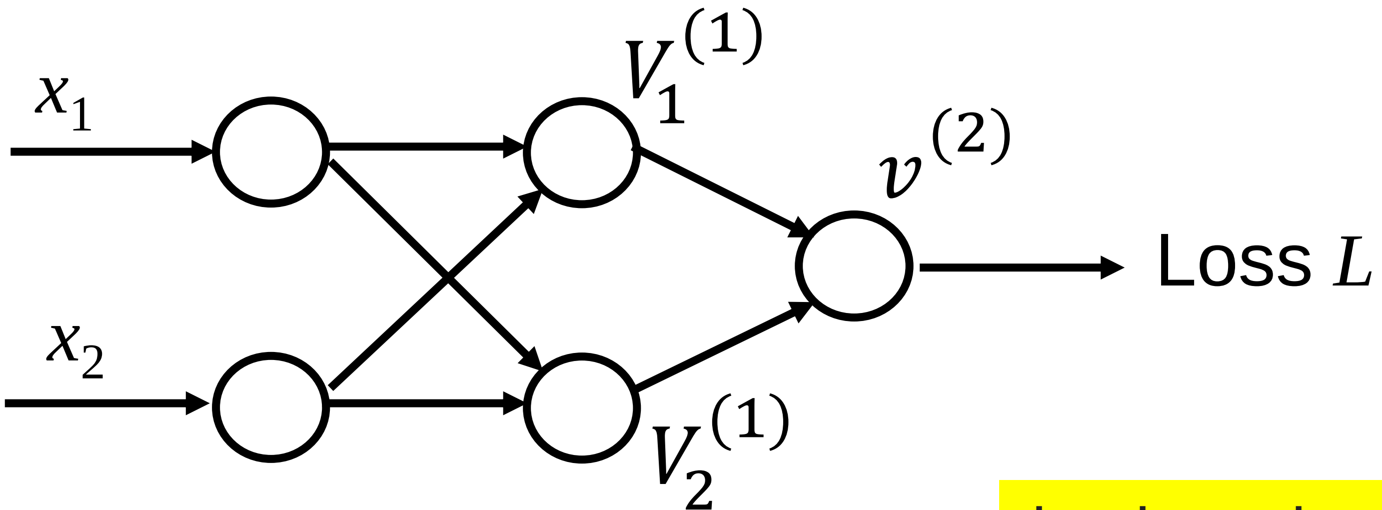
$$\bar{v}^{(2)} = \frac{\partial L}{\partial v^{(2)}} = \frac{2}{|T|}\left(v^{(2)} - T_{true}\right)$$

$$\bar{w}^{(2)} = \left(\bar{w}_1^{(2)}, \bar{w}_2^{(2)}\right) = \frac{\partial L}{\partial w^{(2)}} = \frac{\partial L}{\partial v^{(2)}} \frac{\partial v^{(2)}}{\partial w^{(2)}} = \bar{v}^{(2)} \frac{\partial v^{(2)}}{\partial w^{(2)}}$$

AD in Machine Learning

training of a neural network

Neural network with two linear layers



backward mode

$$\bar{w}^{(2)} = \left(\bar{w}_1^{(2)}, \bar{w}_2^{(2)}\right) = \frac{\partial L}{\partial w^{(2)}} = \frac{\partial L}{\partial v^{(2)}} \frac{\partial v^{(2)}}{\partial w^{(2)}} = \bar{v}^{(2)} \frac{\partial v^{(2)}}{\partial w^{(2)}}$$

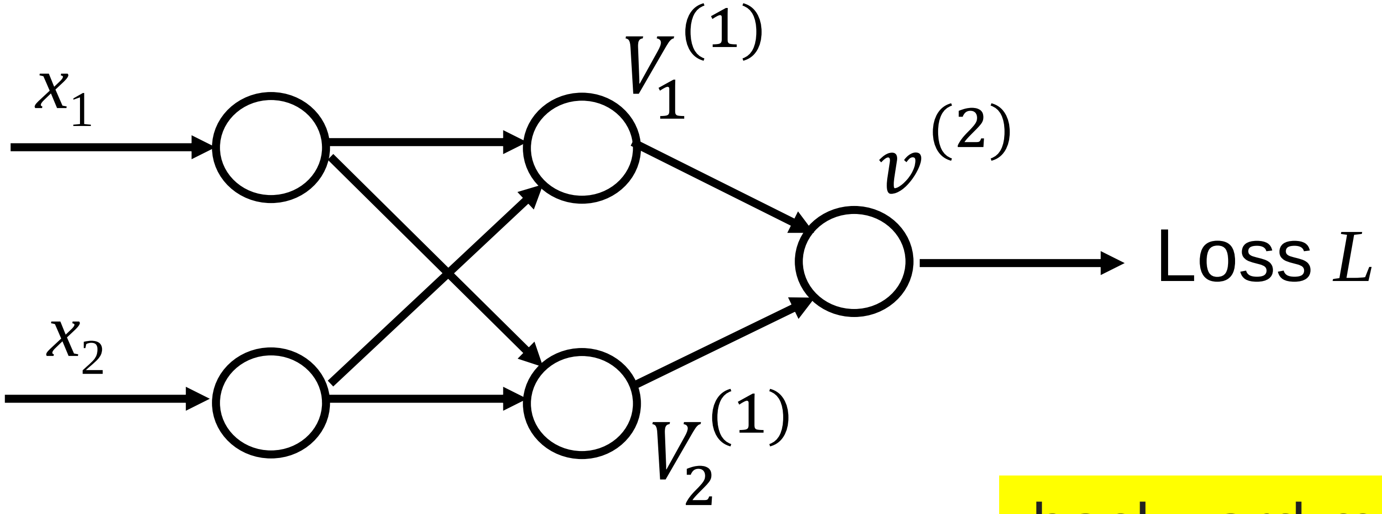
$$\bar{w}^{(2)} = \left(\bar{w}_1^{(2)}, \bar{w}_2^{(2)}\right) = \bar{v}^{(2)} V^{(1)}$$

$$v^{(2)} = w^{(2)T} V^{(1)} + b^{(2)}$$

AD in Machine Learning

training of a neural network

Neural network with two linear layers



backward mode

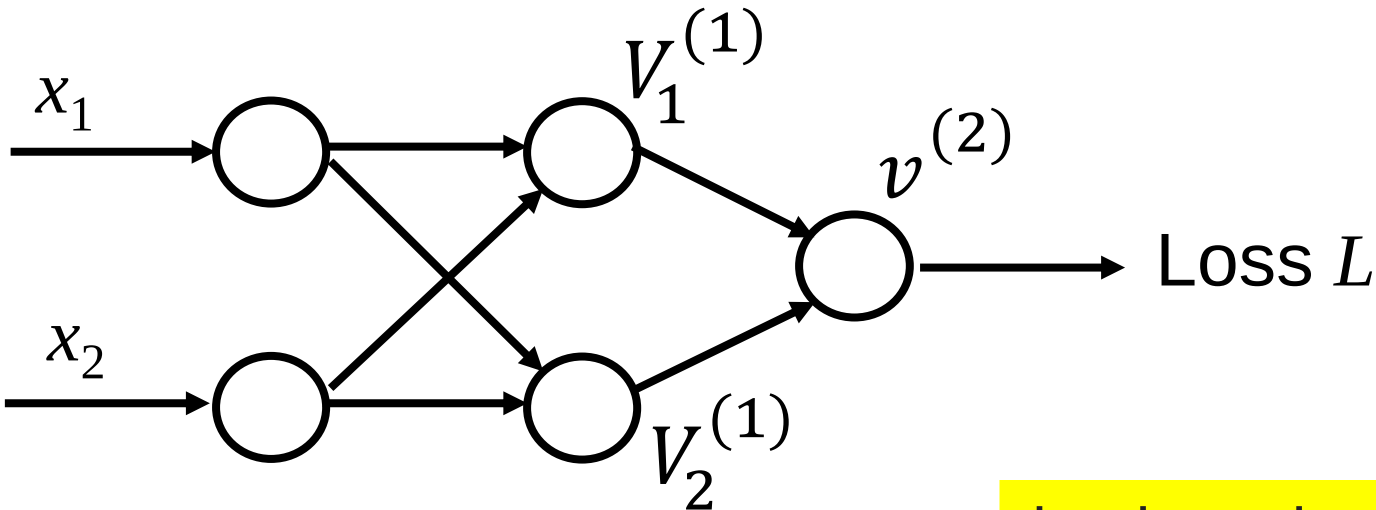
$$\bar{b}^{(2)} = \frac{\partial L}{\partial b^{(2)}} = \frac{\partial L}{\partial v^{(2)}} \frac{\partial v^{(2)}}{\partial b^{(2)}} = \bar{v}^{(2)} \frac{\partial v^{(2)}}{\partial b^{(2)}} = \bar{v}^{(2)}$$

$$v^{(2)} = w^{(2)T} V^{(1)} + b^{(2)}$$

AD in Machine Learning

training of a neural network

Neural network with two linear layers



backward mode

$V^{(1)} = W^{(1)}x + b^{(1)}$

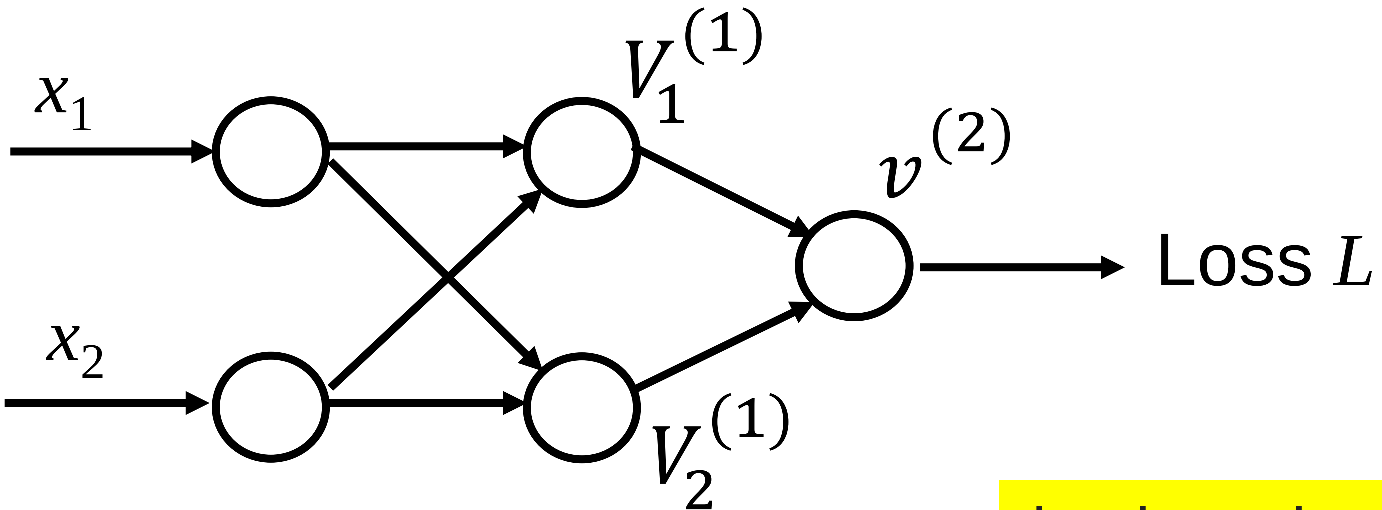
$$\bar{V}^{(1)} = \frac{\partial L}{\partial V^{(1)}} = \bar{v}^{(2)} w^{(2)}$$

$$\bar{W}^{(1)} = \begin{pmatrix} \bar{W}_{11}^{(1)} & \bar{W}_{12}^{(1)} \\ \bar{W}_{21}^{(1)} & \bar{W}_{22}^{(1)} \end{pmatrix} = \frac{\partial L}{\partial W^{(1)}} = \frac{\partial L}{\partial V^{(1)}} \frac{\partial V^{(1)}}{\partial W^{(1)}} = \bar{v}^{(2)} w^{(2)} x$$

AD in Machine Learning

training of a neural network

Neural network with two linear layers



backward mode

$V^{(1)} = W^{(1)}x + b^{(1)}$

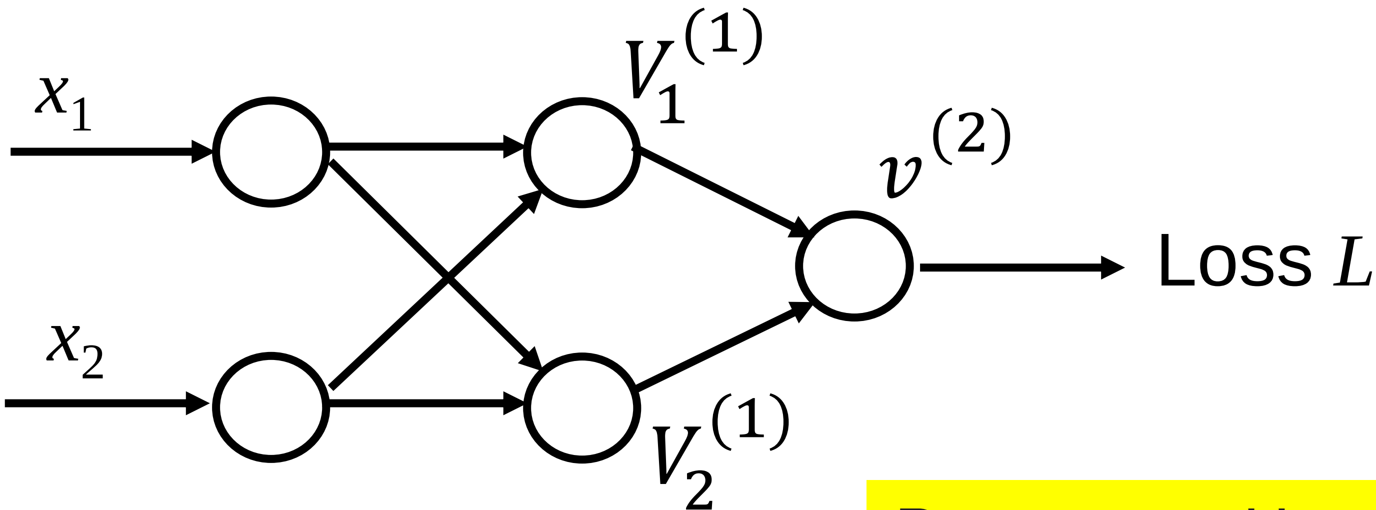
$$\bar{V}^{(1)} = \frac{\partial L}{\partial V^{(1)}} = \bar{v}^{(2)} w^{(2)}$$

$$\bar{b}^{(1)} = \left(\bar{b}_1^{(1)}, \bar{b}_2^{(1)}\right) = \frac{\partial L}{\partial b^{(1)}} = \frac{\partial L}{\partial V^{(1)}} \frac{\partial V^{(1)}}{\partial b^{(1)}} = \bar{v}^{(2)} w^{(2)}$$

AD in Machine Learning

training of a neural network

Neural network with two linear layers



Parameters Upgrade in backward mode (GD)

$$w_{new}^{(2)} = w_{old}^{(2)} - \alpha \bar{w}^{(2)}$$

$$b_{new}^{(2)} = b_{old}^{(2)} - \alpha \bar{b}^{(2)}$$

$$W_{new}^{(1)} = W_{old}^{(1)} - \alpha \bar{W}^{(1)} \qquad b_{new}^{(1)} = b_{old}^{(1)} - \alpha \bar{b}^{(1)}$$