Maths for Computer Science
Calculus

Prof. Magnus Bordewich

Automatic differentiation

Jacobian matrix

We have a function $\boldsymbol{f}: \mathbb{R}^n \to \mathbb{R}^m$ where $\boldsymbol{f} = (f_1(\boldsymbol{x}), f_2(\boldsymbol{x}), \dots, f_m(\boldsymbol{x}))$.

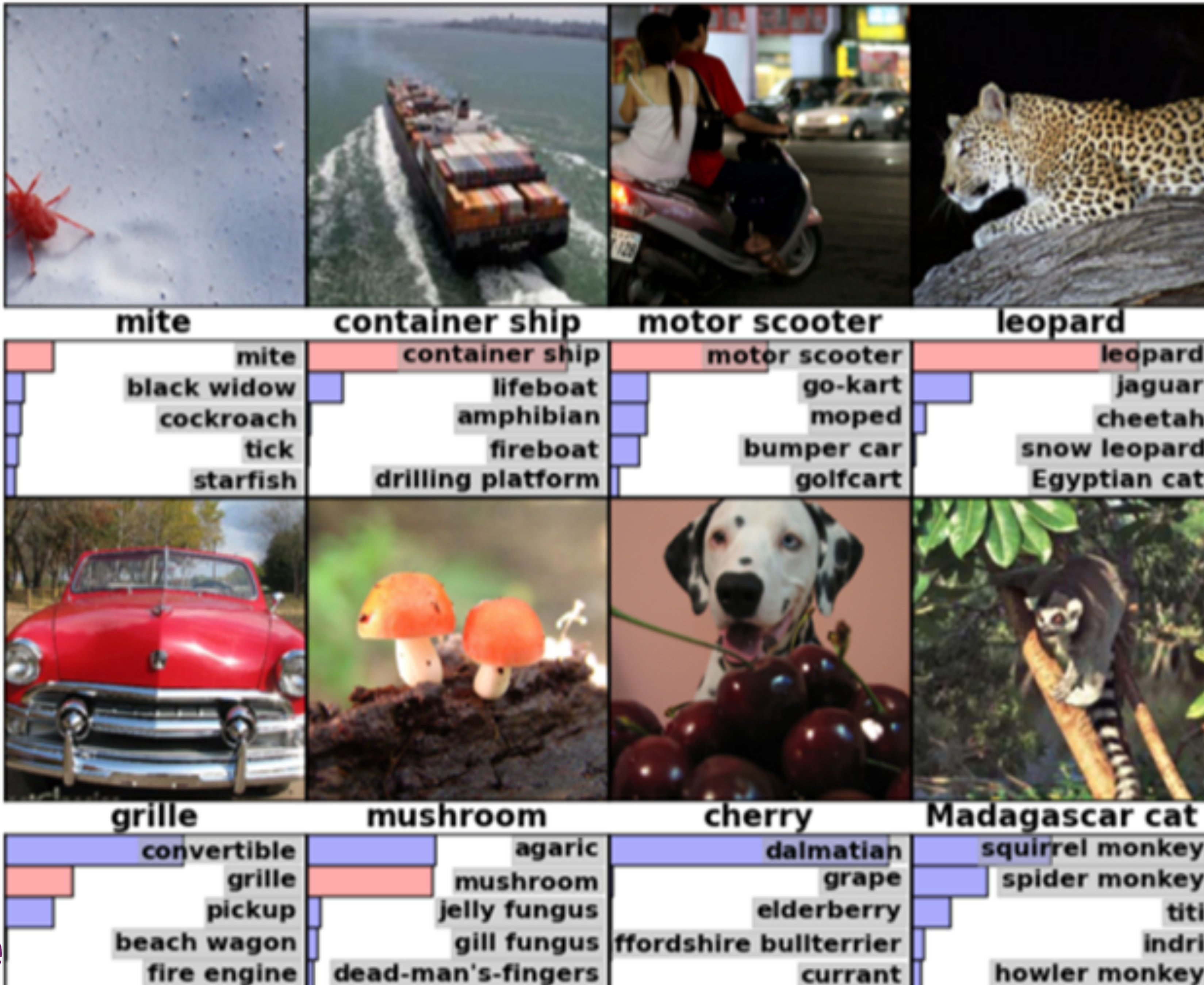
Set $\boldsymbol{J}_{ij} = \dfrac{\partial f_i}{\partial x_j}$.

The resulting **matrix of partial derivatives** is called the **Jacobian matrix**:

$$\mathbf{J} = \begin{pmatrix} \frac{\partial f_1}{\partial x_1} & \cdots & \frac{\partial f_1}{\partial x_n} \\ \vdots & \ddots & \vdots \\ \frac{\partial f_m}{\partial x_1} & \cdots & \frac{\partial f_m}{\partial x_n} \end{pmatrix}$$

So the i^{th} row of the Jacobian matrix of $\boldsymbol{f}$ is the gradient ∇f_i (transposed).

And the j^{th} column is the derivative of $\boldsymbol{f}$ w.r.t. x_j: $\dfrac{\partial \boldsymbol{f}}{\partial x_j}$.

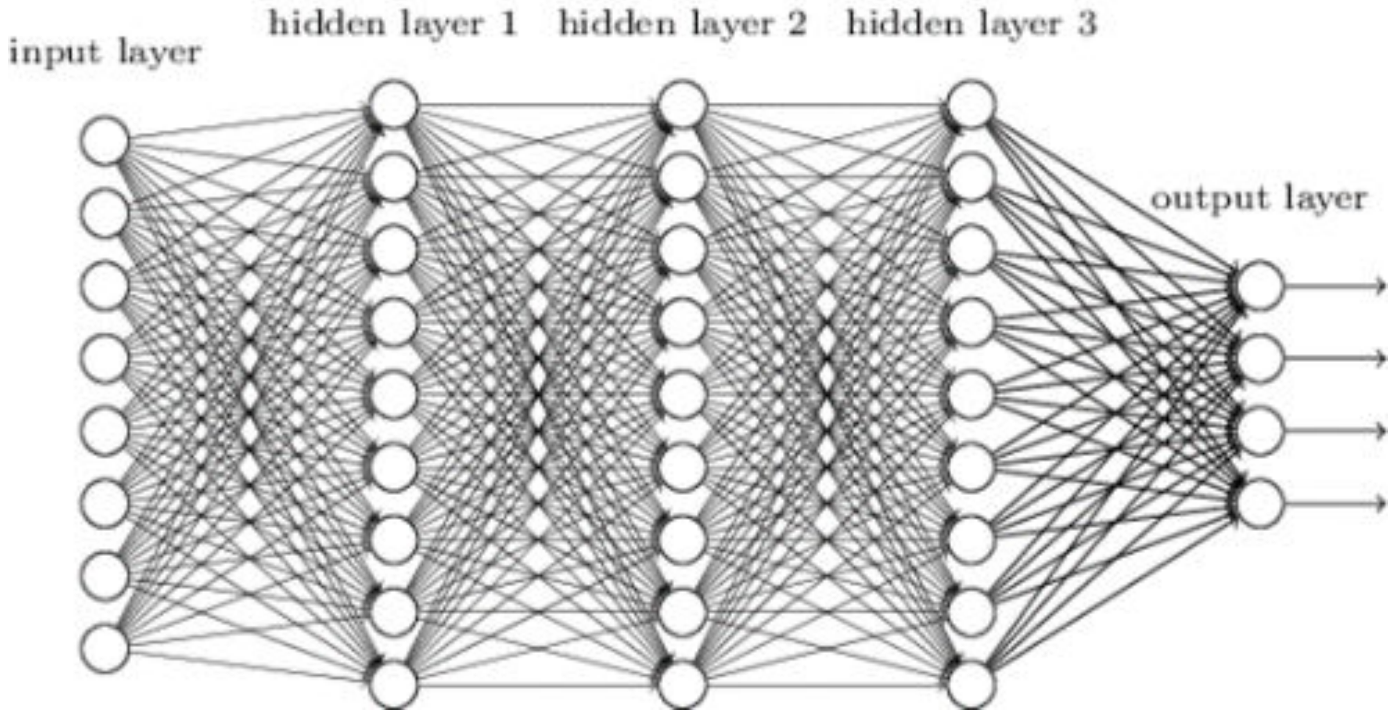

mite
mite
black widow
cockroach
tick
starfish
container ship
container ship
lifeboat
amphibian
fireboat
drilling platform
motor scooter
motor scooter
go-kart
moped
bumper car
golfcart
leopard
leopard
jaguar
cheetah
snow leopard
Egyptian cat
grille
convertible
grille
pickup
beach wagon
fire engine
mushroom
agaric
mushroom
jelly fungus
gill fungus
dead-man's-fingers
cherry
dalmatian
grape
elderberry
ffordshire bullterrier
currant
Madagascar cat
squirrel monkey
spider monkey
titi
indri
howler monkey

Alexnet deep learning neural network

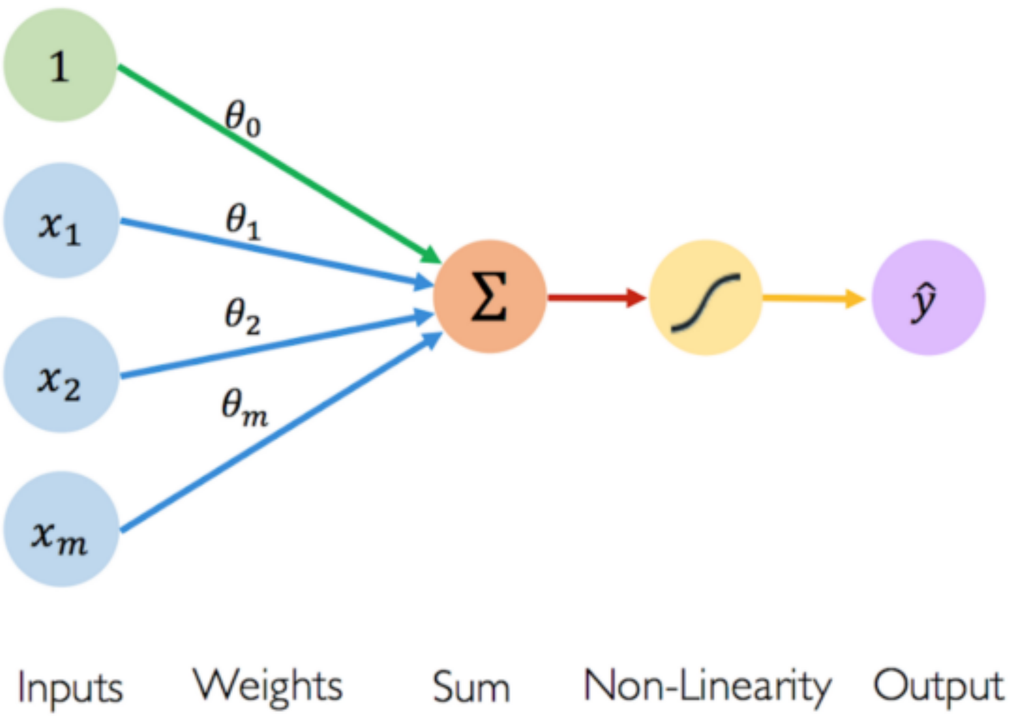
Input is an image: 224x224 pixels x3 channels (RGB), so 150528 values in [0,255]

Outputs are nouns, e.g. Lion, Tiger, Horse etc, where the output vector $\boldsymbol{O} = (o_1, o_2, \ldots, o_{21841})$, gives a match score to each category ($o_i \in [0,1]$).

In between is a neural network with 60 million parameters:



An artificial neuron

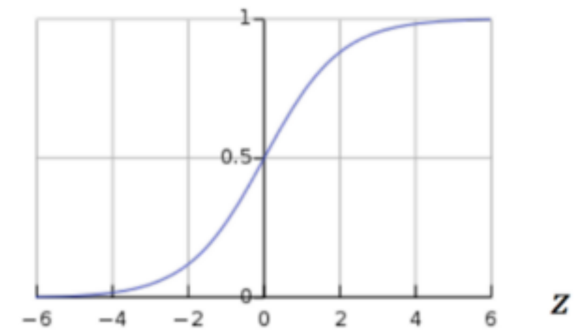


Activation Functions

$$\hat{y} = g\left(\theta_0 + \boldsymbol{X}^T \boldsymbol{\theta} \right)$$

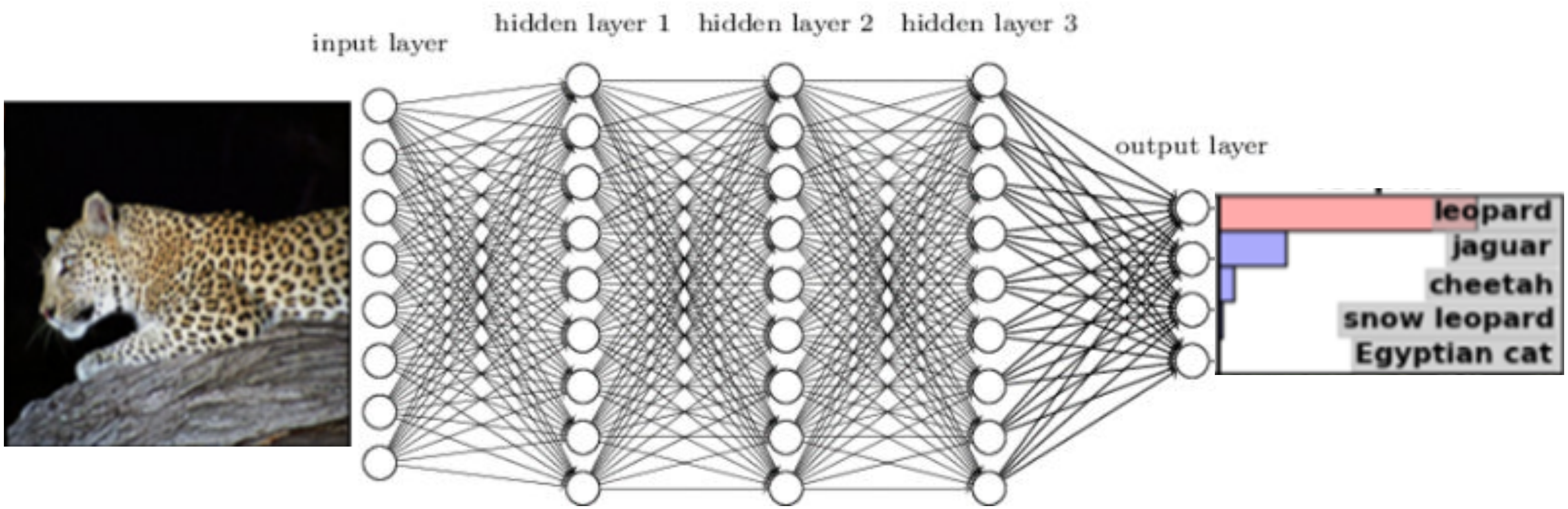
- Example: sigmoid function

$$g(z) = \sigma(z) = \frac{1}{1 + e^{-z}}$$



MIT: Alexander Amini, 2018 introtodeeplearning.com

Learning



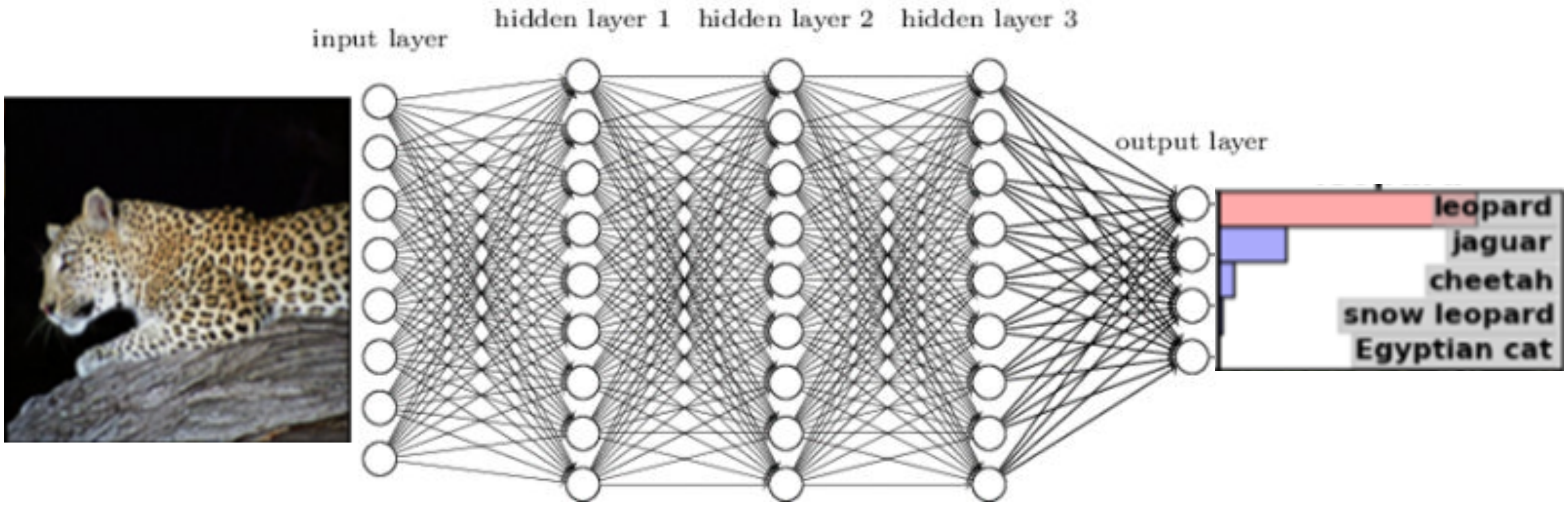
Error
-0.2
+0.2
+0.05
+0.02
+0.0

Total: 0.47

1. Give the network an input for which we know the correct output.
2. Compute the network's output vector.
3. Compute the error relative to the ground truth.
4. Adjust the 60-million parameters a tiny bit to reduce the error.

So we have a function $f\colon \mathbb{R}^{60000000} \to \mathbb{R}^{21841}$ and we want to know what direction to move our parameter vector in to get the greatest reduction in total error: we need ∇f.

Learning



Error
-0.2
+0.2
+0.05
+0.02
+0.0

Total: 0.47

So we have a function $f: \mathbb{R}^{60000000} \to \mathbb{R}$ and we want to know what direction to move our parameter vector in to get the greatest reduction in total error: we need ∇f.

Unfortunately f is a large and complex function – but at least it is made up of the composition of many simpler functions!

Solution: use Automatic Differentiation to compute the gradient!

Computational differentiation

Symbolic differentiation:

Apply mathematical rules to generate closed form solutions.

Problem: combinatorial explosion of terms.

Numerical differentiation:

Estimate derivative from limit formula: $\frac{\partial f}{\partial x_i}(\boldsymbol{x}) \approx \frac{f(\boldsymbol{x}+h.\boldsymbol{e_i})-f(\boldsymbol{x})}{h}$ for small h.

Problem: inaccurate for such large systems.

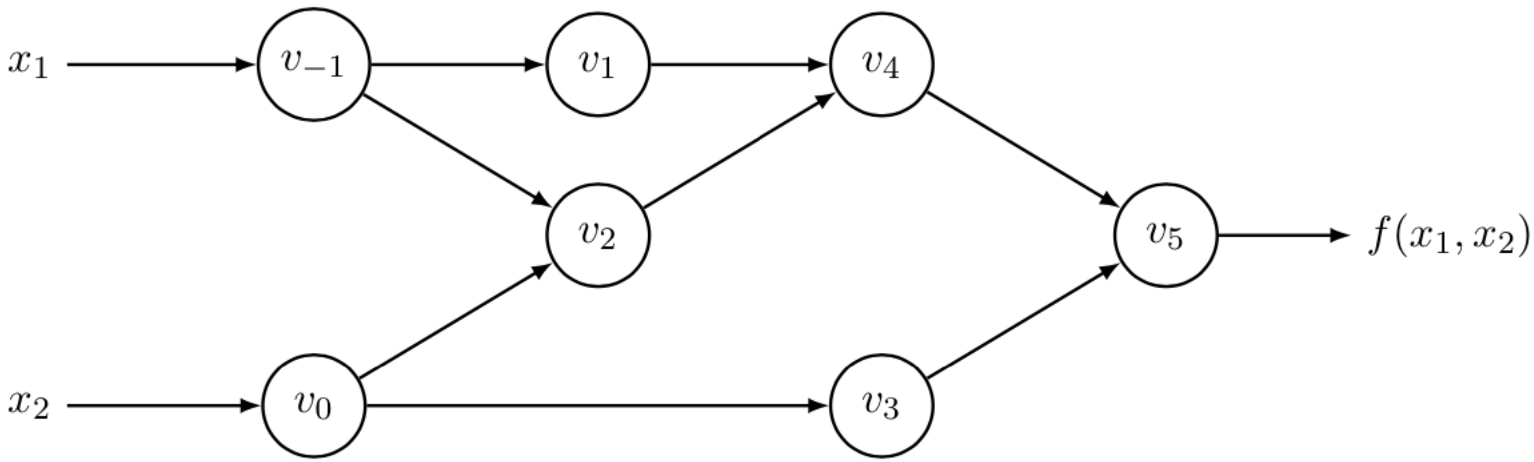
Automatic differentiation:

Efficient and exact.

Durham University

Step 1: create a computation graph from atomic operations

Example: $y(x_1, x_2) = \ln(x_1) + x_1 x_2 - \sin(x_2)$ at point (2,5)



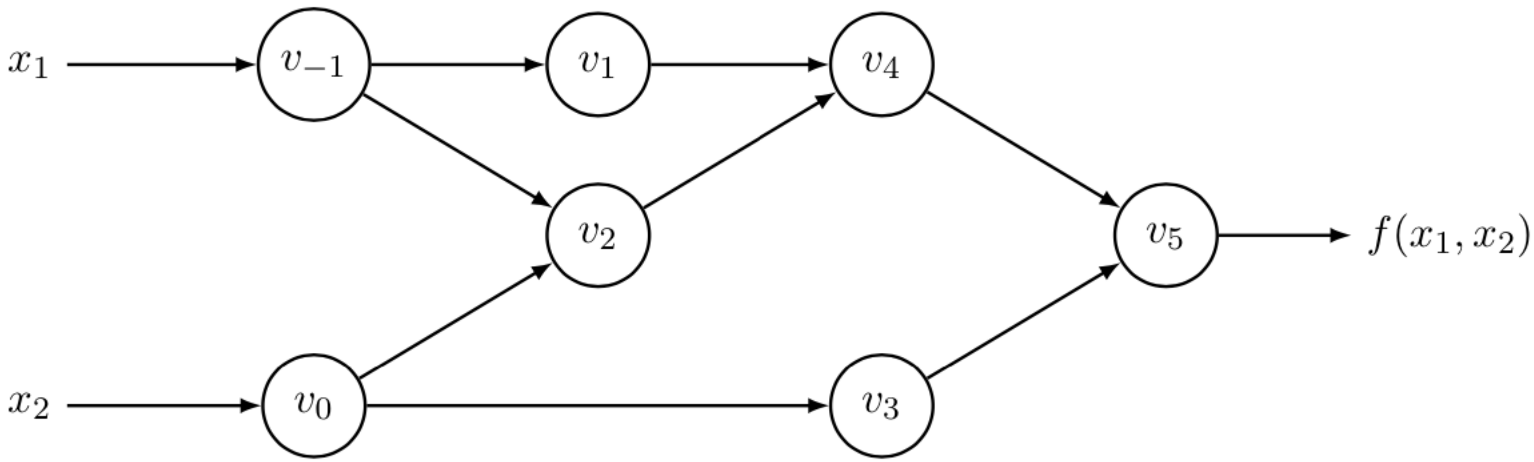
Forward Primal Trace

$$
\begin{aligned}
v_{-1} &= x_1 \\
v_0 &= x_2 \\
\hline
v_1 &= \ln v_{-1} \\
v_2 &= v_{-1} \times v_0 \\
v_3 &= \sin v_0 \\
v_4 &= v_1 + v_2 \\
v_5 &= v_4 - v_3 \\
\hline
y &= v_5
\end{aligned}
$$

Durham University

Step 2: Evaluate the function with a forward pass

Example: $y(x_1, x_2) = \ln(x_1) + x_1 x_2 - \sin(x_2)$ at point (2,5)

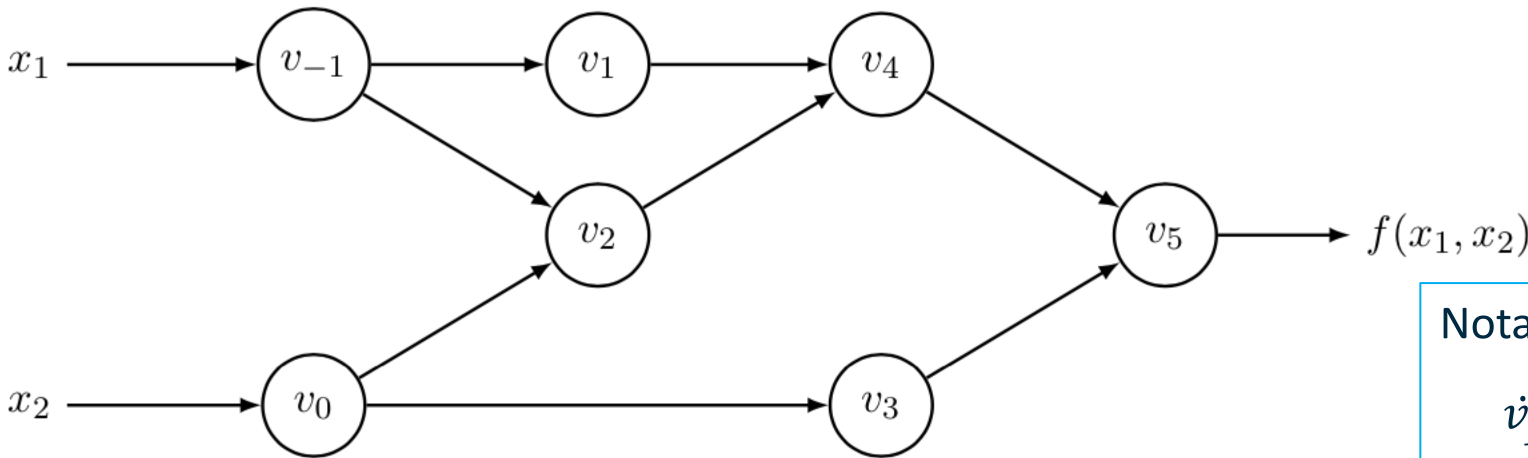


Forward Primal Trace

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

Step 3: Compute partial derivatives of atomic functions

Example: $y(x_1, x_2) = \ln(x_1) + x_1 x_2 - \sin(x_2)$ at point (2,5)



Notation:

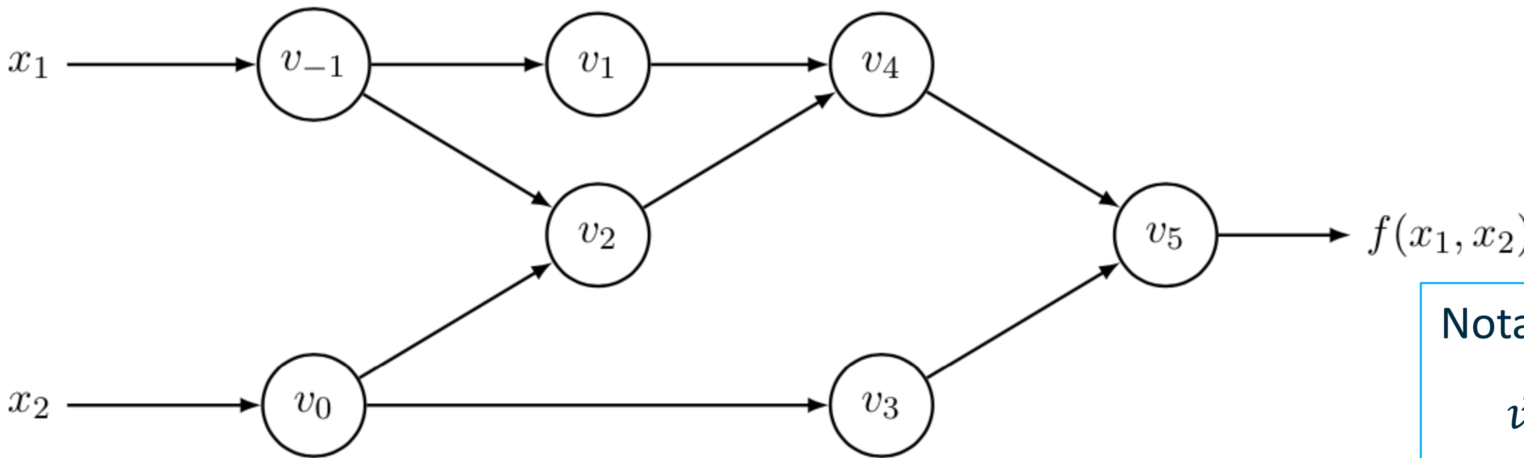
$$\dot{v}_j = \frac{\partial v_j}{\partial x_1}$$

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	
$\dot{v}_{-1}$	$= \dot{x}_1$
$\dot{v}_0$	$= \dot{x}_2$
$\dot{v}_1$	$= \dot{v}_{-1} / v_{-1}$
$\dot{v}_2$	$= \dot{v}_{-1} \times v_0 + \dot{v}_0 \times v_{-1}$
$\dot{v}_3$	$= \dot{v}_0 \times \cos v_0$
$\dot{v}_4$	$= \dot{v}_1 + \dot{v}_2$
$\dot{v}_5$	$= \dot{v}_4 - \dot{v}_3$
$\dot{\boldsymbol{y}}$	$= \dot{\boldsymbol{v}}_5$

Step 4a: Evaluate derivative $\frac{\partial y}{\partial x_i}$ by setting $\dot{x}_i = 1$ others $= 0$.

Example: $y(x_1, x_2) = \ln(x_1) + x_1 x_2 - \sin(x_2)$ at point (2,5)



Notation:

$$\dot{v}_j = \frac{\partial v_j}{\partial x_i}$$

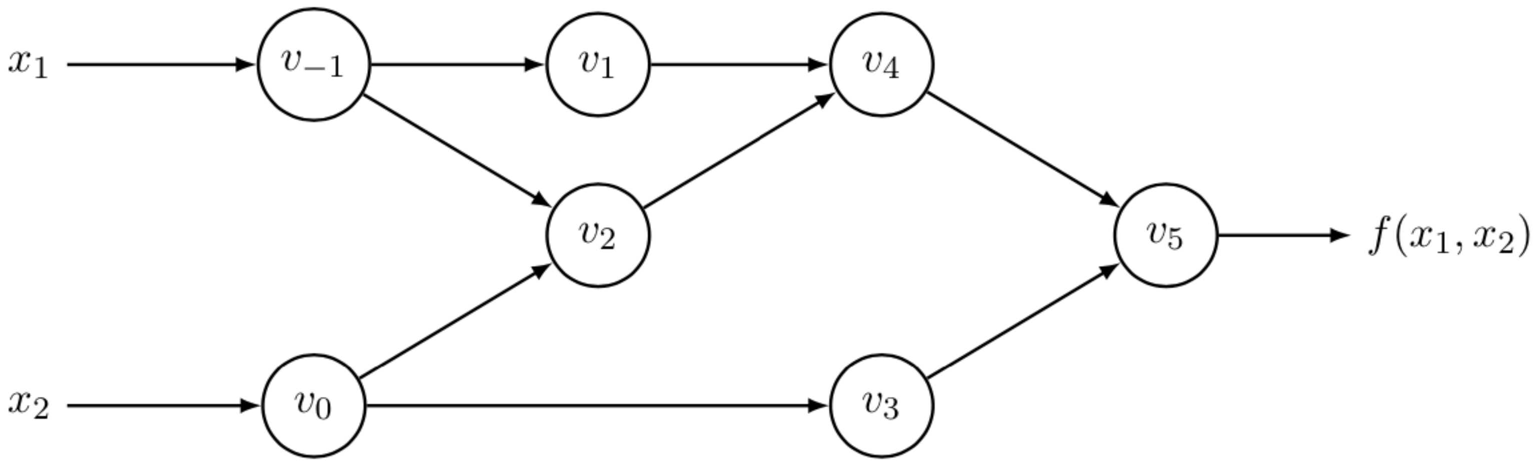
Forward Primal Trace		
v_{-1}	$= x_1$	$= 2$
v_0	$= x_2$	$= 5$
v_1	$= \ln v_{-1}$	$= \ln 2$
v_2	$= v_{-1} \times v_0$	$= 2 \times 5$
v_3	$= \sin v_0$	$= \sin 5$
v_4	$= v_1 + v_2$	$= 0.693 + 10$
v_5	$= v_4 - v_3$	$= 10.693 + 0.959$
y	$= v_5$	$= 11.652$

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

$$\frac{\partial y}{\partial x_1}(\mathbf{2, 5})$$

Step 4b: Evaluate directional derivative in direction (2,1)

Example: $y(x_1, x_2) = \ln(x_1) + x_1 x_2 - \sin(x_2)$ at point (2,5)



Forward Tangent (Derivative)

$$
\begin{aligned}
\dot{v}_{-1} &= \dot{x}_1 \\
\dot{v}_0 &= \dot{x}_2 \\
\hline
\dot{v}_1 &= \dot{v}_{-1}/v_{-1} \\
\dot{v}_2 &= \dot{v}_{-1} \times v_0 + \dot{v}_0 \times v_{-1} \\
\dot{v}_3 &= \dot{v}_0 \times \cos v_0 \\
\dot{v}_4 &= \dot{v}_1 + \dot{v}_2 \\
\dot{v}_5 &= \dot{v}_4 - \dot{v}_3 \\
\hline
\dot{\boldsymbol{y}} &= \dot{\boldsymbol{v}}_5
\end{aligned}
$$

Unit vector in direction (2,1) is $\left(\frac{2}{\sqrt{5}}, \frac{1}{\sqrt{5}}\right)$.

Set $\dot{x_1} = \frac{2}{\sqrt{5}}, \dot{x_2} = \frac{1}{\sqrt{5}}$.

$$\dot{v_1} = \frac{2}{\sqrt{5}} . \frac{1}{2} = \frac{1}{\sqrt{5}}$$

$$\dot{v_2} = \frac{2}{\sqrt{5}} . 5 + \frac{1}{\sqrt{5}} . 2 = \frac{12}{\sqrt{5}}$$

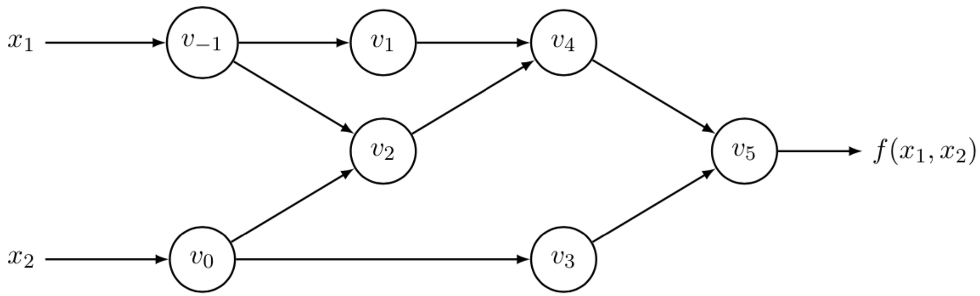
$$\dot{v_3} = \frac{1}{\sqrt{5}} . \cos 5$$

$$\dot{v_4} = \frac{1}{\sqrt{5}} + \frac{12}{\sqrt{5}} = \frac{13}{\sqrt{5}}$$

$$\dot{v_5} = \frac{13}{\sqrt{5}} - \frac{\cos 5}{\sqrt{5}} \approx 5.69$$

Step 5: Reverse mode AD for ∇y

Example: $y(x_1, x_2) = \ln(x_1) + x_1x_2 - \sin(x_2)$ at point (2,5)



Forward Primal Trace

v_{-1}	$= x_1$	:
v_0	$= x_2$	:
v_1	$= \ln v_{-1}$	:
v_2	$= v_{-1} \times v_0$	:
v_3	$= \sin v_0$	:
v_4	$= v_1 + v_2$	:
v_5	$= v_4 - v_3$	:
y	$= v_5$	:

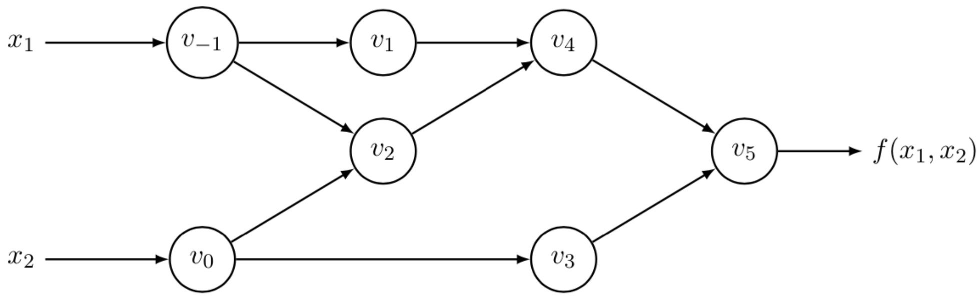
Notation: $\overline{v_i} = \dfrac{\partial y}{\partial v_i}$

Computation:

- Start with $v_5 = y$, so $\overline{v_5} = \dfrac{\partial y}{\partial v_5} = 1$.
- Work backwards through the previous computation adding in the contribution of each variable as it occurs using the chain rule.
- $\overline{v_4} = \dfrac{\partial y}{\partial v_4} = \dfrac{\partial y}{\partial v_5}\dfrac{\partial v_5}{\partial v_4}$ since v_4 only contributes to v_5.
- $$\bar{v}_0 = \bar{v}_2 \frac{\partial v_2}{\partial v_0} + \bar{v}_3 \frac{\partial v_3}{\partial v_0}$$ since v_0 contributes to v_2 and v_3.

Step 5: Reverse mode AD for ∇y

Example: $y(x_1, x_2) = \ln(x_1) + x_1x_2 - \sin(x_2)$ at point (2,5)



Notation:

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

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 Adjoint (Derivative) Trace

$\nabla y(2,5)$

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

Forward mode vs Reverse mode AD

Forward mode: when $f: \mathbb{R}^n \to \mathbb{R}$ one pass can compute the directional derivative.

In general one pass can compute the Jacobian vector product $J_f \boldsymbol{v}$ without have to compute the Jacobian matrix at all!

(Note: the vector $J_f \boldsymbol{v}$ has size 60,000,000 which is OK, the matrix J_f has size 3,600,000,000,000,000 which is not ok.)

But forward mode requires n passes to compute the full gradient ∇f.

Reverse mode computes ∇f in one pass!