Calculus: Jacobian & Hesian Matrices

What is it? first and 2nd derivatives generalized to high dimensions. Necessary to understand backprogation and aptimization landscapes.

$f'(x)$ & $f''(x)$ operating in single dimension. While NN. operate on vectors of millions of dimensions.

Thus when $f'(x)$ & $f''(x)$ are generalized to such vectors we get:

1) Jacobian: matrix of all 1st partial derivatives. Generalizes gradient to vector functions.
2) Hessian: matrix of all 2nd partial derivatives. Captures curvature info.

Essentially a multivariable version of a derivative. Derivative in 2D space gives you slope while gradient in #D gives slope and direction.

Table of Derivatives and how they change with Dimension.

Input	Output	Derivative	Shape/Dimension
Scalar(x)	Scalar (y)	Derivative	1×1
Vector(x)	Scalar (y)	Gradient	$N \times 1$
Vector(x)	Vector ~~Scalar~~(y)	Jacobian	$m \times N$
Vector (x)	Scalar(y)	Hessian	$N \times N$

Jacobian

For function $f: \mathbb{R}^N \rightarrow \mathbb{R}^M$ (N input, m outputs).
Jacobian is an $m \times N$ matrix:

$$J = \begin{bmatrix} \frac{\partial f_1}{\partial x_1} & \frac{\partial f_1}{\partial x_2} & \cdots \\ \frac{\partial f_2}{\partial x_1} & \frac{\partial f_2}{\partial x_2} & \frac{\partial f_m}{\partial x_N} \\ & \cdots & \end{bmatrix}$$

row i: how output i changes with each input.
column j: how each output changes with input j

Example: $\mathbb{R}^2 \to \mathbb{R}^2 : f(x, y) = \begin{bmatrix} x^2 + y \\ xy \end{bmatrix}$

① $\frac{\partial f^1}{\partial x} = 2x$ $\frac{\partial f^1}{\partial y} = 1$

$\frac{\partial f^2}{\partial x} = y$ $\frac{\partial f^2}{\partial y} = x$

② $J = \begin{bmatrix} 2x & 1 \\ y & x \end{bmatrix}$

③ $J_{(2,3)} = \begin{bmatrix} 4 & 1 \\ 3 & 2 \end{bmatrix}$

Hessian Matrix

For a scalar valued function; $f : \mathbb{R}^N \to \mathbb{R}$ (like a loss function).

The hessian is the $N \times N$ matrix of 2nd order partial derivatives.

$$H = \nabla^2 f = \begin{bmatrix} \frac{\partial^2 f}{\partial x_1 \partial x_1} & \frac{\partial^2 f}{\partial x_1 \partial x_2} \\ \frac{\partial^2 f}{\partial x_2 \partial x_1} & \frac{\partial^2 f}{\partial x_2 \partial x_2} \cdots \end{bmatrix}$$

- Hessian captures how gradient itself changes, curvature.
- Symmetric, for continuous 2nd order partials:

$$\frac{\partial^2 f}{\partial x_i \partial x_j} = \frac{\partial^2 f}{\partial x_j \partial x_i})$$

so $H = H^T$

Example:

loss function, $f(x, y) = x^2 + 3xy + y^2$:

① $\nabla f = \begin{bmatrix} 2x + 3y \\ 3x + 2y \end{bmatrix}$

② $\frac{\partial^2 f}{\partial x^2} = 2$ $\frac{\partial^2 f}{\partial x \partial y} = 3$

$\frac{\partial^2 f}{\partial y \partial x} = 3$ $\frac{\partial^2 f}{\partial y^2} = 2$

③ $H = \begin{bmatrix} 2 & 3 \\ 3 & 2 \end{bmatrix}$

Critical points: at a critical point (gradient/derivative = 0)
the Heissan's eigenvalues tell us the nature of critical point:

1) All positive: function curves UP in all directions. Local min.
2) All negative: function curves DOWN in all directions. Local max.
3) Mixed signs: curves up in some, down in others. Saddle point.

So, why does this matter? In high dimension NNs knowing the loss landscape, saddle points are far more common than other points and understanding when that happens helps explain why optimization stalls.

Why does Jacobian & Hessian Matrices even matter?

ML Applications

1. Backpropagation = Jacobian-Vector products: when computing gradients in NN each layer contributes a Jacobian: $\nabla_x J = J_1^T J_2^T \cdots J_n^T \nabla_y L$

2. Hessian-Free Optimization: some algorithms use Hessian info. without computing the full matrix. Conjugate gradient methods can compute Hessian-vector products efficiently.

J.M.

Basically, 1. Jacobian matrix is absolute foundation of backpropagation, but in reality only Jacobian-vector products are used to compute gradients because J.M. are too big.

2. Hessian used to determine curvature of loss surface, allowing optimization methods to adjust step sizes and directions better.