Day 20: Jacobians: The Multidimensional Calculus of Machine Learning

How Vector-Valued Derivatives Power Transformations and Learning in AI

1. Beyond Scalars: The Need for Jacobians

In Day 19, we explored gradients for scalar-valued functions $f : \mathbb{R}^n \to \mathbb{R}$. But in AI, we frequently encounter vector-valued functions $\mathbf{F} : \mathbb{R}^n \to \mathbb{R}^m$ where:

$$\mathbf{F}(\mathbf{x}) = \begin{bmatrix} f_1(\mathbf{x}) \\ f_2(\mathbf{x}) \\ \vdots \\ f_m(\mathbf{x}) \end{bmatrix}$$

For such functions, we need the **Jacobian matrix** $J_{\mathbf{F}}$ to capture how each output component changes with respect to each input component.

2. Formal Definition of the Jacobian Matrix

For a differentiable function $\mathbf{F} : \mathbb{R}^n \to \mathbb{R}^m$, the Jacobian matrix is defined as:

$$J_{\mathbf{F}}(\mathbf{x}) = \begin{bmatrix} \frac{\partial f_1}{\partial x_1} & \frac{\partial f_1}{\partial x_2} & \cdots & \frac{\partial f_1}{\partial x_n} \\ \frac{\partial f_2}{\partial x_1} & \frac{\partial f_2}{\partial x_2} & \cdots & \frac{\partial f_2}{\partial x_n} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial f_m}{\partial x_1} & \frac{\partial f_m}{\partial x_2} & \cdots & \frac{\partial f_m}{\partial x_n} \end{bmatrix}$$

This $m \times n$ matrix contains all first-order partial derivatives of the vector-valued function.

3. A Detailed Example: Understanding the Jacobian

Let's analyze the function $\mathbf{F} : \mathbb{R}^2 \to \mathbb{R}^2$ defined by:

$$\mathbf{F}(x, y) = \begin{bmatrix} x^2 + y \\ xy \end{bmatrix}$$

The Jacobian matrix is:

$$J_{\mathbf{F}}(x, y) = \begin{bmatrix} \frac{\partial}{\partial x}(x^2 + y) & \frac{\partial}{\partial y}(x^2 + y) \\ \frac{\partial}{\partial x}(xy) & \frac{\partial}{\partial y}(xy) \end{bmatrix} = \begin{bmatrix} 2x & 1 \\ y & x \end{bmatrix}$$

Interpreting the Jacobian at Specific Points

- At $(1, 2)$: $J_{\mathbf{F}}(1, 2) = \begin{bmatrix} 2 & 1 \\ 2 & 1 \end{bmatrix}$
  - This tells us how small changes in (x, y) affect the output
  - Increasing x by a small amount Δx changes f_1 by $2\Delta x$ and f_2 by $2\Delta x$
  - Increasing y by a small amount Δy changes both f_1 and f_2 by Δy
- At $(0, 1)$: $J_{\mathbf{F}}(0, 1) = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}$
  - Changing x doesn't affect f_1 but affects f_2
  - Changing y affects f_1 but not f_2

4. The Jacobian as a Linear Approximation

Just as the derivative provides a linear approximation for scalar functions, the Jacobian provides the best linear approximation for vector-valued functions:

$$\mathbf{F}(\mathbf{x} + \Delta\mathbf{x}) \approx \mathbf{F}(\mathbf{x}) + J_{\mathbf{F}}(\mathbf{x})\Delta\mathbf{x}$$

This approximation is crucial in optimization, where we take small steps based on local gradient information.

5. The Jacobian in Backpropagation

Backpropagation is essentially the chain rule applied to neural networks, and the Jacobian plays a central role. For a composition of functions $\mathbf{F} = \mathbf{G} \circ \mathbf{H}$, the chain rule states:

$$J_{\mathbf{F}}(\mathbf{x}) = J_{\mathbf{G}}(\mathbf{H}(\mathbf{x})) \cdot J_{\mathbf{H}}(\mathbf{x})$$

In neural networks, each layer implements a function, and the overall network is a composition of these functions. Backpropagation efficiently computes the product of Jacobians from output to input.

6. Special Cases of the Jacobian

Important Special Cases

- **Gradient:** When $m = 1$, the Jacobian is a row vector—this is the gradient ∇f
- **Jacobian Determinant:** When $m = n$, the determinant of the Jacobian ($\det J_{\mathbf{F}}$) measures how the function locally scales volumes
- **Identity Jacobian:** If $\mathbf{F}$ is linear, $\mathbf{F}(\mathbf{x}) = A\mathbf{x}$, then $J_{\mathbf{F}}(\mathbf{x}) = A$ (constant)

7. Applications in AI and Machine Learning

Where Jacobians Power Modern AI

- **Neural Network Training:** Backpropagation multiplies Jacobians through the network layers
- **Normalizing Flows:** The Jacobian determinant is used to compute how probability densities change under transformations
- **Computer Vision:** Jacobians describe how image transformations affect pixel values
- **Robotics:** In kinematics, the Jacobian relates joint velocities to end-effector velocities
- **Implicit Layers:** Layers defined by equations (rather than explicit functions) require specialized Jacobian computation
- **Adversarial Robustness:** Jacobians help understand how input perturbations affect outputs

8. The Jacobian Determinant and Volume Change

For functions $\mathbf{F} : \mathbb{R}^n \to \mathbb{R}^n$ (same input and output dimension), the Jacobian determinant $\det J_{\mathbf{F}}(\mathbf{x})$ has a beautiful geometric interpretation: it represents how the function locally scales volumes.

If $\det J_{\mathbf{F}}(\mathbf{x}) > 1$, the function expands volume near $\mathbf{x}$. If $0 < \det J_{\mathbf{F}}(\mathbf{x}) < 1$, it contracts volume. If $\det J_{\mathbf{F}}(\mathbf{x}) < 0$, it also reverses orientation.

This is crucial in normalizing flows for generative modeling, where we need to compute how probability densities transform.

9. Computational Aspects: How Frameworks Compute Jacobians

Modern deep learning frameworks use automatic differentiation to compute Jacobians efficiently:

- **Forward Mode:** Computes one column of the Jacobian at a time (efficient for few outputs)
- **Reverse Mode:** Computes one row of the Jacobian at a time (efficient for few inputs, used in backpropagation)
- **JAX and Jacobian-Vector Products:** Modern frameworks like JAX provide efficient Jacobian-vector product operations

10. Exercises for Understanding

1. For $\mathbf{F}(x,y,z) = \begin{bmatrix} x^2 + yz \\ \sin(xy) \\ e^{x+z} \end{bmatrix}$, compute the Jacobian matrix $J_{\mathbf{F}}(x,y,z)$

2. For the function $\mathbf{F}(x,y) = \begin{bmatrix} x^2 - y^2 \\ 2xy \end{bmatrix}$, compute the Jacobian determinant and interpret its geometric meaning
3. Explain why the chain rule for Jacobians involves matrix multiplication rather than simple scalar multiplication
4. For a simple neural network layer $\mathbf{F}(\mathbf{x}) = \sigma(W\mathbf{x} + \mathbf{b})$ with elementwise sigmoid activation, compute the Jacobian $J_{\mathbf{F}}(\mathbf{x})$

Key Takeaway

The Jacobian matrix is the fundamental tool for understanding and computing derivatives of vector-valued functions. It extends the concept of gradient to multidimensional outputs, providing a complete picture of how each output component changes with respect to each input component. In AI, Jacobians are indispensable for backpropagation through deep networks, analyzing transformations in computer vision and graphics, and implementing normalizing flows for generative modeling. Understanding Jacobians is essential for anyone working with complex, multidimensional transformations in machine learning.