1. Partial Derivatives

How a function changes with respect to one variable while others remain fixed

Formula:

$$\frac{\partial f}{\partial x_i} = \lim_{h \to 0} \frac{f(x_1, \ldots, x_i + h, \ldots, x_n) - f(x_1, \ldots, x_i, \ldots, x_n)}{h}$$

Terms Explained:

- $f(x_1, \ldots, x_n)$: Multivariate function
- Fixed variables: All x_j where $j \neq i$

@AIinMinutes

2. Gradient

Vector of partial derivatives pointing in direction of steepest ascent

Formula:

$$\nabla f(\mathbf{x}) = \begin{pmatrix} \frac{\partial f}{\partial x_1} \\ \frac{\partial f}{\partial x_2} \\ \vdots \\ \frac{\partial f}{\partial x_n} \end{pmatrix}$$

Terms Explained:

- ∇f: Gradient vector
- $\frac{\partial f}{\partial x_i}$: Partial derivative with respect to x_i
- Direction: Points toward greatest increase of f
- Magnitude: Rate of increase in that direction

@AIinMinutes

3. Chain Rule

Method for computing derivatives of composite functions

Formula:

$$\frac{\partial z}{\partial x_i} = \sum_{j=1}^{m} \frac{\partial z}{\partial y_j} \cdot \frac{\partial y_j}{\partial x_i}$$

Terms Explained:

- $z = f(y_1, \ldots, y_m)$: Outer function
- $y_j = g_j(x_1, \ldots, x_n)$: Inner functions
- $\frac{\partial z}{\partial y_j}$: Partial derivative of outer function
- $\frac{\partial y_j}{\partial x_i}$: Partial derivative of inner function

@AIinMinutes

4. Jacobian

Matrix of all first-order partial derivatives of vector-valued function

Formula: For $\mathbf{f} : \mathbb{R}^n \to \mathbb{R}^m$:

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

Terms Explained:

- f_i: The i-th output function
- x_j: The j-th input variable
- $\frac{\partial f_i}{\partial x_j}$: Rate of change of f_i with respect to x_j

@AIinMinutes

5. Outer Product

Operation on two vectors producing a matrix (rank-1 matrices)

Formula:

$$\mathbf{u} \otimes \mathbf{v} = \mathbf{u}\mathbf{v}^T = \begin{pmatrix} u_1 v_1 & u_1 v_2 & \cdots & u_1 v_n \\ u_2 v_1 & u_2 v_2 & \cdots & u_2 v_n \\ \vdots & \vdots & \ddots & \vdots \\ u_m v_1 & u_m v_2 & \cdots & u_m v_n \end{pmatrix}$$

Terms Explained:

- $\mathbf{u} \in \mathbb{R}^m$: Column vector
- $\mathbf{v} \in \mathbb{R}^n$: Column vector
- $\mathbf{v}^T$: Transpose (row vector)
- Result: $m \times n$ matrix

@AIinMinutes

6. Logits

Raw, unnormalized predictions from neural network

Formula (for single layer):

$$\mathbf{z} = \mathbf{W}\mathbf{x} + \mathbf{b}$$

Terms Explained:

- $\mathbf{z} \in \mathbb{R}^n$: Vector of logits
- $\mathbf{W} \in \mathbb{R}^{n \times d}$: Weight matrix
- $\mathbf{x} \in \mathbb{R}^d$: Input features
- $\mathbf{b} \in \mathbb{R}^n$: Bias vector

@AIinMinutes

7. Softmax

Function converting logits into probability distribution

Formula:

$$\text{softmax}(\mathbf{z})_i = \frac{e^{z_i}}{\sum_{j=1}^{n} e^{z_j}}$$

Terms Explained:

- $\mathbf{z} = (z_1, \ldots, z_n)$: Logits vector
- e^{z_i}: Exponential of each logit
- $\sum_{j=1}^{n} e^{z_j}$: Normalization term
- Output: Vector with values in $(0, 1)$ that sum to 1

@AIinMinutes

8. Cross-Entropy

Loss function measuring difference between probability distributions

Formula:

$$H(p, q) = -\sum_{i=1}^{n} p_i \log(q_i)$$

Terms Explained:

- $p = (p_1, \dots, p_n)$: True probability distribution
- $q = (q_1, \dots, q_n)$: Predicted probability distribution
- $\log(q_i)$: Natural logarithm of predicted probability
- Minimal when $p = q$

@AIinMinutes

9. Gradient Descent

Optimization algorithm; GD, Mini-batch GD, and SGD

Formula:

$$\theta_{t+1} = \theta_t - \eta \nabla_\theta J(\theta_t)$$

Terms Explained:

- θ_t: Parameter vector at iteration t
- η: Learning rate
- $\nabla_\theta J(\theta_t)$: Gradient of cost function

@AIinMinutes

10. Adam Optimization

Adaptive optimization algorithm with per-parameter learning rates

Formula:

$$
\begin{aligned}
m_t &= \beta_1 m_{t-1} + (1-\beta_1) g_t \\
v_t &= \beta_2 v_{t-1} + (1-\beta_2) g_t^2 \\
\hat{m}_t &= \frac{m_t}{1-\beta_1^t} \\
\hat{v}_t &= \frac{v_t}{1-\beta_2^t} \\
\theta_t &= \theta_{t-1} - \alpha \cdot \frac{\hat{m}_t}{\sqrt{\hat{v}_t} + \epsilon}
\end{aligned}
$$

Terms Explained:

- g_t: Gradient at time t
- m_t, v_t: First and second moment estimates
- β_1, β_2: Exponential decay rates (typically 0.9, 0.999)
- α: Learning rate
- ϵ: Small constant for numerical stability