# Comprehensive Machine Learning Math Cheatsheet

## Contents



## 1 Linear Algebra

- **Dot Product**:
$$\mathbf{a} \cdot \mathbf{b} = \sum_{i=1}^{n} a_i b_i$$
- **Matrix Multiplication**: For matrices $\mathbf{A} \in \mathbb{R}^{m \times k}$, $\mathbf{B} \in \mathbb{R}^{k \times n}$, element $C_{ij}$ of $\mathbf{C} = \mathbf{AB}$ is
$$C_{ij} = \sum_{k} A_{ik} B_{kj}$$
- **Transpose**: For matrix $\mathbf{A}$, the transpose $\mathbf{A}^T$ has elements
$$(A^T)_{ij} = A_{ji}$$
- **Identity Matrix**: $\mathbf{IA} = \mathbf{A}$, where $\mathbf{I}$ is a square matrix with 1s on the diagonal, 0s elsewhere.
- **Determinant (2x2)**: For $\mathbf{A} = \begin{bmatrix} a & b \\ c & d \end{bmatrix}$,
$$\text{det}(\mathbf{A}) = ad - bc$$
- **Inverse Matrix**: For invertible $\mathbf{A}$, $\mathbf{A}^{-1}\mathbf{A} = \mathbf{I}$.
- **Eigenvalues and Eigenvectors**: For square matrix $\mathbf{A}$, $\mathbf{A}\mathbf{v} = \lambda\mathbf{v}$, where $\lambda$ is an eigenvalue and $\mathbf{v}$ is an eigenvector.
- **Singular Value Decomposition (SVD)**: $\mathbf{A} = \mathbf{U}□\mathbf{V}^T$, where $\mathbf{U}, \mathbf{V}$ are orthogonal, □ is diagonal.

## 2 Calculus

- **Derivative**:
$$\frac{d}{dx}f(x) = \lim_{h \to 0} \frac{f(x+h) - f(x)}{h}$$
- **Power Rule**:
$$\frac{d}{dx}x^n = nx^{n-1}$$
- **Product Rule**:
$$\frac{d}{dx}[f(x)g(x)] = f'(x)g(x) + f(x)g'(x)$$
- **Chain Rule**:
$$\frac{d}{dx}f(g(x)) = f'(g(x)) \cdot g'(x)$$
- **Partial Derivative**: $\frac{\partial f}{\partial x_i}$, treating other variables as constants.
- **Gradient**: $\nabla f = \left(\frac{\partial f}{\partial x_1}, \dots, \frac{\partial f}{\partial x_n}\right)$.
- **Hessian Matrix**: $H_{ij} = \frac{\partial^2 f}{\partial x_i \partial x_j}$.
- **Integral (Fundamental Theorem)**: If $F'(x) = f(x)$, then
$$\int_a^b f(x)\,dx = F(b) - F(a)$$

## 3 Probability & Statistics

- **Expected Value**: For random variable $X$,
$$\mathbb{E}[X] = \sum x_i P(x_i) \text{ (discrete)} \quad \text{or} \quad \int x f(x)\,dx \text{ (continuous)}$$
- **Mean**:
$$\mu = \frac{1}{n}\sum_{i=1}^{n} x_i$$
- **Variance**:
$$\sigma^2 = \frac{1}{n}\sum_{i=1}^{n}(x_i - \mu)^2$$
- **Standard Deviation**:
$$\sigma = \sqrt{\sigma^2}$$
- **Covariance**: For variables $X, Y$,
$$\text{Cov}(X,Y) = \mathbb{E}[(X - \mu_X)(Y - \mu_Y)]$$
- **Bayes' Theorem**:
$$P(A \mid B) = \frac{P(B \mid A)P(A)}{P(B)}$$
- **Normal Distribution PDF**:
$$f(x) = \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{(x-\mu)^2}{2\sigma^2}}$$

- **Entropy**: For discrete distribution $P$,

$$H(P) = -\sum_i P(x_i) \log P(x_i)$$

- **KL Divergence**: For distributions $P, Q$,

$$D_{\text{KL}}(P \parallel Q) = \sum_i P(x_i) \log \frac{P(x_i)}{Q(x_i)}$$

# 4 Optimization

- **Mean Squared Error (MSE) Loss**:

$$L = \frac{1}{n} \sum_{i=1}^{n} (y_i - \hat{y}_i)^2$$

- **Cross-Entropy Loss**: For binary classification,

$$L = -\frac{1}{n} \sum_{i=1}^{n} [y_i \log(\hat{y}_i) + (1 - y_i) \log(1 - \hat{y}_i)]$$

- **Gradient Descent**:

$$\mathbf{w} \leftarrow \mathbf{w} - \alpha \nabla L(\mathbf{w})$$

- **Learning Rate**: $\alpha$ controls step size.
- **Convergence**: $\nabla L(\mathbf{w}) \approx 0$.
- **Momentum**:

$$\mathbf{v} = \beta \mathbf{v} - \alpha \nabla L(\mathbf{w}), \quad \mathbf{w} \leftarrow \mathbf{w} + \mathbf{v}$$

- **Adam Optimizer**: Combines momentum and RMSProp,

$$m_t = \beta_1 m_{t-1} + (1 - \beta_1) \nabla L, \quad v_t = \beta_2 v_{t-1} + (1 - \beta_2)(\nabla L)^2$$

$$\mathbf{w}_t = \mathbf{w}_{t-1} - \alpha \frac{\hat{m}_t}{\sqrt{\hat{v}_t} + \epsilon}$$

- **L1 Regularization**: $L = L_0 + \lambda \sum |w_i|$.
- **L2 Regularization**: $L = L_0 + \lambda \sum w_i^2$.

# 5 Neural Networks

- **Neuron Output**:

$$y = f(\mathbf{W}\mathbf{x} + \mathbf{b})$$

- **Sigmoid Activation**:

$$f(x) = \frac{1}{1 + e^{-x}}$$

- **ReLU Activation**:

$$f(x) = \max(0, x)$$

- **Tanh Activation**:

$$f(x) = \tanh(x) = \frac{e^x - e^{-x}}{e^x + e^{-x}}$$

- **Softmax**: For output $z_i$,

$$\sigma(z_i) = \frac{e^{z_i}}{\sum_j e^{z_j}}$$

- **Backpropagation**: Update weights using $\frac{\partial L}{\partial \mathbf{w}}$ via chain rule.
- **Dropout**: Randomly set fraction $p$ of neurons to 0 during training.
- **Batch Normalization**: Normalize layer inputs,

$$\hat{x}_i = \frac{x_i - \mu_B}{\sqrt{\sigma_B^2 + \epsilon}}, \quad y_i = amma\hat{x}_i + \beta$$

# 6 Clustering and Dimensionality Reduction

- **K-Means Objective**: Minimize

$$J = \sum_{i=1}^{n} \sum_{k=1}^{K} r_{ik} \|\mathbf{x}_i - \mu_k\|^2$$

  where $r_{ik}$ is 1 if $\mathbf{x}_i$ belongs to cluster $k$, else 0.
- **PCA (Principal Component Analysis)**: Project data onto top $k$ eigenvectors of covariance matrix ▯.
- **t-SNE Objective**: Minimize KL divergence between high-dimensional and low-dimensional distributions.