





Sarwan Ali

Department of Computer Science  
Georgia State University

 The Mathematical Foundation of Intelligence 

# Today's Mathematical Journey

- 1 Linear Algebra: The Language of Data
- 2 Probability & Statistics: Handling Uncertainty
- 3 Calculus: The Mathematics of Change
- 4 Putting It All Together: ML Applications
- 5 Practical Tips & Next Steps

## Think of ML as a Recipe

### Traditional Cooking:

- Follow instructions step by step
- Measure ingredients
- Apply heat and time
- Get consistent results

### Machine Learning:

- **Linear Algebra**: Organize ingredients (data)
- **Probability**: Handle uncertainty in cooking
- **Calculus**: Optimize the recipe automatically
- Get intelligent results!

### Key Point

Math gives us the language to describe and solve learning problems systematically!

# Linear Algebra: Why It's Everywhere in ML

**Data is just numbers organized in arrays!**

**Real-world data as vectors/matrices:**

- **Images:** Pixel intensity arrays
- **Text:** Word frequency vectors
- **Audio:** Wave amplitude sequences
- **Customer data:** Feature vectors

**Example: House Price Prediction**

$$\text{House} = \begin{bmatrix} 2000 \\ 3 \\ 2 \\ 1995 \end{bmatrix}$$

(sq ft, bedrooms, bathrooms, year built)

**Bottom Line**

Linear algebra lets us manipulate ALL this data efficiently with simple operations!

# Vectors: The Building Blocks

## What is a Vector?

- A list of numbers
- Represents a point in space
- Or a direction and magnitude

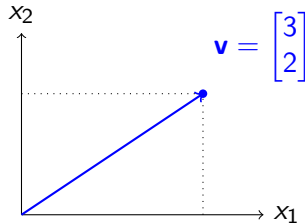
## Notation:

$$\mathbf{v} = \begin{bmatrix} v_1 \\ v_2 \\ \vdots \\ v_n \end{bmatrix}$$

## Example:

$$\text{student} = \begin{bmatrix} 85 \\ 92 \\ 78 \\ 88 \end{bmatrix}$$

(Math, Science, English, History grades)



## Vector Operations:

- Addition:  $\mathbf{a} + \mathbf{b}$
- Scalar multiplication:  $c\mathbf{a}$
- Dot product:  $\mathbf{a} \cdot \mathbf{b}$

# Essential Vector Operations

## 1. Dot Product (Inner Product):

$$\mathbf{a} \cdot \mathbf{b} = \sum_{i=1}^n a_i b_i = a_1 b_1 + a_2 b_2 + \cdots + a_n b_n$$

**Geometric meaning:** Measures similarity/correlation

## 2. Vector Norm (Length):

$$\|\mathbf{v}\| = \sqrt{\mathbf{v} \cdot \mathbf{v}} = \sqrt{v_1^2 + v_2^2 + \cdots + v_n^2}$$

## 3. Unit Vector:

$$\hat{\mathbf{v}} = \frac{\mathbf{v}}{\|\mathbf{v}\|}$$

## ML Connection

### Similarity in Recommendations:

User A:  $\begin{bmatrix} 5 \\ 3 \\ 1 \\ 4 \end{bmatrix}$  (movie ratings)

User B:  $\begin{bmatrix} 4 \\ 3 \\ 2 \\ 5 \end{bmatrix}$

$$\text{Similarity} = \frac{\mathbf{A} \cdot \mathbf{B}}{\|\mathbf{A}\| \|\mathbf{B}\|}$$

Higher similarity  $\rightarrow$  Better recommendations!

# Matrices: Collections of Vectors

## What is a Matrix?

- 2D array of numbers
- Collection of vectors
- Represents datasets or transformations

## Notation:

$$\mathbf{A} = \begin{bmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ a_{21} & a_{22} & \cdots & a_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ a_{m1} & a_{m2} & \cdots & a_{mn} \end{bmatrix}$$

Size:  $m \times n$  (rows  $\times$  columns)

## Dataset Example

Student grades matrix:

$$\mathbf{Grades} = \begin{bmatrix} 85 & 92 & 78 \\ 90 & 88 & 95 \\ 76 & 84 & 82 \\ 88 & 91 & 89 \end{bmatrix}$$

- Rows = Students
- Columns = Subjects
- Each row is a student vector
- Each column is a subject vector

# Matrix Operations That Power ML (Alternative)

## 1. Matrix Multiplication: $\mathbf{AB} = \mathbf{C}$

$$\mathbf{A}_{2 \times 2} = \begin{bmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{bmatrix} \quad \mathbf{B}_{2 \times 2} = \begin{bmatrix} b_{11} & b_{12} \\ b_{21} & b_{22} \end{bmatrix} \quad = \mathbf{C}_{2 \times 2} = \begin{bmatrix} c_{11} & c_{12} \\ c_{21} & c_{22} \end{bmatrix}$$

Where:  $c_{ij} = \sum_k a_{ik} b_{kj}$

## 2. Matrix Transpose: $\mathbf{A}^T$ (flip rows and columns)

## 3. Matrix Inverse: $\mathbf{A}^{-1}$ (undoes the transformation)

## ML Application

Linear regression:  $\mathbf{w} = (\mathbf{X}^T \mathbf{X})^{-1} \mathbf{X}^T \mathbf{y}$  (finds best fit line!)



# Eigenvalues & Eigenvectors: Finding Special Directions

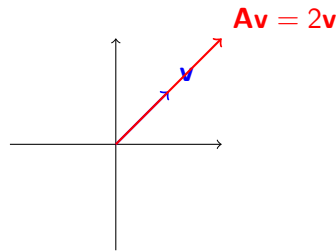
**The Big Idea:** Some vectors don't change direction when transformed by a matrix!

$$\mathbf{A}\mathbf{v} = \lambda\mathbf{v}$$

- $\mathbf{v}$ : eigenvector (special direction)
- $\lambda$ : eigenvalue (scaling factor)

## Why Care?

- Reveals the "natural" directions in data
- Used in dimensionality reduction (PCA)
- Helps understand data structure
- Critical for many ML algorithms



Same direction, different length!

## PCA Example

Find directions of maximum variance in data to reduce dimensions while preserving information.

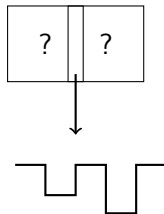
## Real world is messy and uncertain!

### Why Probability in ML?

- Data has noise and errors
- Predictions are uncertain
- Models make assumptions
- Need to quantify confidence

### Examples:

- "70% chance of rain"
- "95% confidence interval"
- "Spam probability: 0.85"



Probability Distribution

### Key Insight

ML algorithms learn probability distributions from data to make informed predictions!

# Probability Basics

## Probability Rules:

- $0 \leq P(A) \leq 1$  (probabilities are between 0 and 1)
- $P(\text{certain event}) = 1$
- $P(\text{impossible event}) = 0$
- $P(A \text{ or } B) = P(A) + P(B) - P(A \text{ and } B)$

## Conditional Probability:

$$P(A|B) = \frac{P(A \text{ and } B)}{P(B)}$$

"Probability of A given B has occurred"

## Bayes' Theorem:

$$P(A|B) = \frac{P(B|A)P(A)}{P(B)}$$

## Spam Detection

- $P(\text{spam} | \text{"free money"}) = ?$
- Given email contains "free money", what's probability it's spam?
- Use Bayes' theorem with training data!

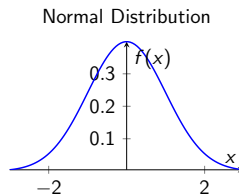
$$P(\text{spam} | \text{words}) = \frac{P(\text{words} | \text{spam})P(\text{spam})}{P(\text{words})}$$

# Important Probability Distributions

## 1. Normal (Gaussian) Distribution:

$$f(x) = \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{(x-\mu)^2}{2\sigma^2}}$$

- Bell-shaped curve
- Characterized by mean  $\mu$  and variance  $\sigma^2$
- Central Limit Theorem
- Used everywhere in ML!



## 2. Bernoulli Distribution:

- Binary outcomes (0 or 1)
- Probability  $p$  of success
- Models coin flips, binary classification

### ML Applications

- **Normal:** Linear regression errors, feature distributions
- **Bernoulli:** Logistic regression, binary classification

# Statistics: Making Sense of Data

## Descriptive Statistics:

- **Mean:**  $\bar{x} = \frac{1}{n} \sum_{i=1}^n x_i$
- **Median:** Middle value when sorted
- **Mode:** Most frequent value
- **Variance:**  $\sigma^2 = \frac{1}{n} \sum_{i=1}^n (x_i - \bar{x})^2$
- **Standard Deviation:**  $\sigma = \sqrt{\sigma^2}$

## Why Important?

- Understand your data first!
- Detect outliers and anomalies
- Choose appropriate algorithms

## Inferential Statistics:

- **Hypothesis Testing:** Is this result significant?
- **Confidence Intervals:** Range of plausible values
- **p-values:** Evidence against null hypothesis
- **Correlation:** Linear relationship strength

## A/B Testing

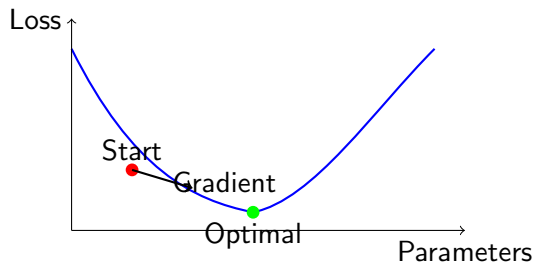
- Test two versions of ML model
- Measure performance difference
- Use statistics to determine if difference is significant
- Make data-driven decisions!

## How do machines learn?

They optimize! And optimization needs calculus.

### The Learning Process:

- 1 Start with random model
- 2 Measure how wrong it is (loss function)
- 3 Find direction to improve
- 4 Take a step in that direction
- 5 Repeat until optimal



# Derivatives: The Rate of Change

## What is a Derivative?

$$f'(x) = \lim_{h \rightarrow 0} \frac{f(x+h) - f(x)}{h}$$

- Slope of the tangent line
- Rate of change at a point
- Direction of steepest increase

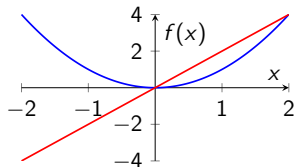
## Common Derivatives:

$$\frac{d}{dx}[x^n] = nx^{n-1}$$

$$\frac{d}{dx}[e^x] = e^x$$

$$\frac{d}{dx}[\ln x] = \frac{1}{x}$$

$$\frac{d}{dx}[\sin x] = \cos x$$



## ML Connection

### Gradient Descent:

- (1) • Calculate derivative of loss function
- (2) • Move in opposite direction (negative gradient)
- (3) • Minimize error automatically!

$$(4) \quad w_{new} = w_{old} - \alpha \frac{\partial \text{Loss}}{\partial w}$$

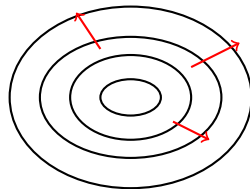
# Partial Derivatives & Gradients: Multivariable Optimization

## Partial Derivatives:

For function  $f(x, y)$ :

$$\frac{\partial f}{\partial x} = \text{rate of change w.r.t. } x \quad (5)$$

$$\frac{\partial f}{\partial y} = \text{rate of change w.r.t. } y \quad (6)$$



## Gradient Vector:

$$\nabla f = \begin{bmatrix} \frac{\partial f}{\partial x} \\ \frac{\partial f}{\partial y} \end{bmatrix}$$

- Points in direction of steepest increase
- Magnitude = rate of steepest increase
- Perpendicular to level curves

Gradient points away from minimum

## Neural Networks

**Backpropagation** uses chain rule to compute gradients of loss w.r.t. all parameters efficiently!



# Chain Rule: The Heart of Backpropagation

**Chain Rule:** If  $y = f(g(x))$ , then:

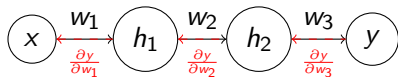
$$\frac{dy}{dx} = \frac{dy}{dg} \cdot \frac{dg}{dx}$$

**Multiple Variables:** If  $z = f(x, y)$  where  $x = g(t)$  and  $y = h(t)$ :

$$\frac{dz}{dt} = \frac{\partial z}{\partial x} \frac{dx}{dt} + \frac{\partial z}{\partial y} \frac{dy}{dt}$$

**Why Critical for ML?**

- Neural networks are compositions  $f_n(f_{n-1}(\dots f_1(x)))$
- Need gradients w.r.t. all parameters
- Chain rule propagates error backwards



## Backpropagation

$$\frac{\partial \text{Loss}}{\partial w_1} = \frac{\partial \text{Loss}}{\partial y} \cdot \frac{\partial y}{\partial h_2} \cdot \frac{\partial h_2}{\partial h_1} \cdot \frac{\partial h_1}{\partial w_1}$$

# Optimization: Finding the Best Solution

## Optimization Problem:

$$\min_w f(w)$$

Where  $f(w)$  is the loss/cost function

### Critical Points:

- $\nabla f(w) = 0$  (gradient is zero)
- Could be minimum, maximum, or saddle point

### Second Derivative Test:

- $f''(w) > 0$ : Local minimum
- $f''(w) < 0$ : Local maximum
- $f''(w) = 0$ : Inconclusive

## Gradient Descent Algorithm:

- 1 Initialize parameters  $w_0$
- 2 For  $t = 0, 1, 2, \dots$ :
  - Compute gradient:  $g_t = \nabla f(w_t)$
  - Update:  $w_{t+1} = w_t - \alpha g_t$
- 3 Stop when converged

### Learning Rate

- $\alpha$  too large: Overshooting
- $\alpha$  too small: Slow convergence
- Need to tune carefully!

# Example 1: Linear Regression

**Problem:** Predict house prices from features

## Linear Algebra:

- Data matrix  $\mathbf{X}$  (houses  $\times$  features)
- Target vector  $\mathbf{y}$  (prices)
- Parameter vector  $\mathbf{w}$  (weights)
- Prediction:  $\hat{\mathbf{y}} = \mathbf{X}\mathbf{w}$

## Statistics:

- Assume errors  $\sim$  Normal distribution
- Least squares estimation
- R-squared for model evaluation

## Calculus:

- Loss function:  $L(\mathbf{w}) = \|\mathbf{X}\mathbf{w} - \mathbf{y}\|^2$
- Take derivative:  $\frac{\partial L}{\partial \mathbf{w}} = 2\mathbf{X}^T(\mathbf{X}\mathbf{w} - \mathbf{y})$
- Set to zero:  $\mathbf{X}^T(\mathbf{X}\mathbf{w} - \mathbf{y}) = 0$
- Solve:  $\mathbf{w} = (\mathbf{X}^T\mathbf{X})^{-1}\mathbf{X}^T\mathbf{y}$

## All Three Together!

Linear algebra for data representation, statistics for assumptions, calculus for optimization.

## Example 2: Logistic Regression

**Problem:** Classify emails as spam or not spam

**The Model:**

$$P(\text{spam}|\mathbf{x}) = \frac{1}{1 + e^{-\mathbf{w}^T \mathbf{x}}}$$

**Linear Algebra:**

- Feature vectors  $\mathbf{x}$  (word counts, etc.)
- Weight vector  $\mathbf{w}$
- Linear combination  $\mathbf{w}^T \mathbf{x}$

**Probability:**

- Sigmoid function maps to  $[0,1]$
- Bernoulli distribution for binary outcome
- Maximum likelihood estimation

**Calculus Optimization:**

Loss function (negative log-likelihood):

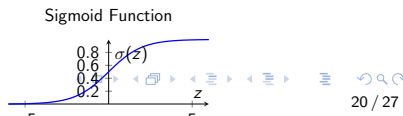
$$L(\mathbf{w}) = - \sum_{i=1}^n [y_i \log p_i + (1 - y_i) \log(1 - p_i)]$$

Gradient:

$$\frac{\partial L}{\partial \mathbf{w}} = \mathbf{X}^T (\mathbf{p} - \mathbf{y})$$

Update rule:

$$\mathbf{w}_{t+1} = \mathbf{w}_t - \alpha \mathbf{X}^T (\mathbf{p} - \mathbf{y})$$



## Example 3: Principal Component Analysis (PCA)

**Problem:** Reduce data dimensions while preserving information

### Mathematical Foundation:

#### Linear Algebra:

- Eigendecomposition:  $\mathbf{C}\mathbf{v} = \lambda\mathbf{v}$
- Matrix multiplication for projection
- Orthogonal transformation

#### Statistics:

- Variance measures information content
- Covariance shows feature relationships
- Principal components capture maximum variance

### The Algorithm:

- 1 Center the data:  $\mathbf{X}_c = \mathbf{X} - \bar{\mathbf{X}}$
- 2 Compute covariance:  
$$\mathbf{C} = \frac{1}{n}\mathbf{X}_c^T\mathbf{X}_c$$
- 3 Find eigenvalues/eigenvectors of  $\mathbf{C}$
- 4 Sort by eigenvalue magnitude
- 5 Keep top  $k$  eigenvectors as principal components
- 6 Project data:  $\mathbf{Y} = \mathbf{X}_c\mathbf{W}_k$

### Applications

- Image compression, Data visualization
- Feature extraction, Noise reduction

## Example 4: Neural Networks - The Full Package

### Neural Networks Use ALL Mathematical Concepts!

#### Forward Pass (Linear Algebra):

$$\mathbf{z}^{(1)} = \mathbf{W}^{(1)}\mathbf{x} + \mathbf{b}^{(1)} \quad (7)$$

$$\mathbf{a}^{(1)} = \sigma(\mathbf{z}^{(1)}) \quad (8)$$

$$\mathbf{z}^{(2)} = \mathbf{W}^{(2)}\mathbf{a}^{(1)} + \mathbf{b}^{(2)} \quad (9)$$

$$\mathbf{y} = \text{softmax}(\mathbf{z}^{(2)}) \quad (10)$$

#### Probability:

- Softmax outputs probabilities
- Cross-entropy loss
- Dropout for regularization
- Bayesian neural networks

#### Backward Pass (Calculus):

$$\frac{\partial L}{\partial \mathbf{W}^{(2)}} = \frac{\partial L}{\partial \mathbf{z}^{(2)}} (\mathbf{a}^{(1)})^T \quad (11)$$

$$\frac{\partial L}{\partial \mathbf{a}^{(1)}} = (\mathbf{W}^{(2)})^T \frac{\partial L}{\partial \mathbf{z}^{(2)}} \quad (12)$$

$$\frac{\partial L}{\partial \mathbf{W}^{(1)}} = \frac{\partial L}{\partial \mathbf{z}^{(1)}} \mathbf{x}^T \quad (13)$$

#### Chain Rule Everywhere:

$$\frac{\partial L}{\partial \mathbf{W}^{(1)}} = \frac{\partial L}{\partial \mathbf{y}} \frac{\partial \mathbf{y}}{\partial \mathbf{z}^{(2)}} \frac{\partial \mathbf{z}^{(2)}}{\partial \mathbf{a}^{(1)}} \frac{\partial \mathbf{a}^{(1)}}{\partial \mathbf{z}^{(1)}} \frac{\partial \mathbf{z}^{(1)}}{\partial \mathbf{W}^{(1)}}$$

## The Beauty of Neural Networks

They seamlessly combine linear algebra (efficient computation), probability (uncertainty handling), and calculus (automatic optimization) into one powerful framework!

# How to Master These Mathematical Concepts

## Study Strategy:

- 1 **Start with intuition** - Understand the "why"
- 2 **Practice calculations** - Build mechanical skills
- 3 **Code implementations** - Solidify understanding
- 4 **Apply to ML problems** - See connections
- 5 **Teach others** - Test your knowledge

## Don't Fear the Math!

- You don't need to prove theorems
- Focus on understanding and application
- Use computational tools when appropriate

## Recommended Resources:

### Books:

- "Mathematics for Machine Learning" - Deisenroth et al.
- "Linear Algebra Done Right" - Axler
- "Introduction to Statistical Learning" - James et al.

### Online:

- Khan Academy (basics)
- 3Blue1Brown (visual explanations)
- MIT OpenCourseWare
- Coursera/edX math courses

### Practice:

- Implement algorithms from scratch

# Common Misconceptions & How to Avoid Them

**Misconception 1:** "I need to memorize all formulas"

**Reality:** Understanding concepts  $\neq$  memorizing formulas. Focus on intuition and derive when needed.

**Misconception 2:** "Linear algebra is just matrix arithmetic"

**Reality:** It's about understanding transformations, spaces, and geometric relationships.

**Misconception 3:** "Statistics is just calculating means and variances"

**Reality:** It's about reasoning under uncertainty and making valid inferences from data.

**Misconception 4:** "Calculus is just taking derivatives"

**Reality:** It's about understanding rates of change and optimization in high dimensions.

**Remember: Math is a tool to solve problems, not an end in itself!**



# Your Mathematical Journey in Machine Learning

## Mathematical Foundations

Linear Algebra  
Probability & Statistics  
Calculus

## Basic ML Algorithms

Linear/Logistic Regression  
k-Means Clustering  
Decision Trees

## Advanced Algorithms

Neural Networks  
SVMs, Random Forests  
Deep Learning

## Research & Innovation

Novel Architectures  
Theoretical Analysis  
Published Research

### Timeline Estimate:

- Foundations: 2-3 months
- Basic ML: 2-3 months
- Advanced: 6-12 months
- Research: Ongoing!

### Key Success Factors:

- Consistent daily practice
- Balance theory with implementation
- Join study groups/communities
- Work on real projects

# Summary: The Mathematical Toolkit for ML

**You now have the roadmap to mathematical mastery in ML!**



## Linear Algebra

- Data representation
- Efficient computation
- Transformations
- Dimensionality reduction



## Probability & Statistics

- Handle uncertainty
- Model assumptions
- Inference and testing
- Confidence quantification



## Calculus

- Optimization
- Gradient descent
- Backpropagation
- Parameter tuning


## Final Message


Mathematics is not a barrier to machine learning - it's your superpower! Start with one concept, practice consistently, and watch as the entire ML landscape opens up before you.


**Questions? Let's dive deeper into the mathematics of intelligence!**

# Thank You!

Mathematical Prerequisites for Machine Learning

 [sali85@student.gsu.edu]

 Department of Computer Science

 Georgia State University

*"The only way to learn mathematics is to do mathematics."  
- Paul Halmos*

**Ready to start your mathematical journey in ML?**