Understanding Machine Learning Mathematics: The Why and How Behind Every Operation

Research Team

September 29, 2025

Abstract

This document provides detailed explanations of the mathematical operations in machine learning algorithms, focusing on **why** each operation is necessary and **how** it achieves its purpose. Through concrete examples with actual numerical computations, we demonstrate the inner workings of algorithms and the reasoning behind each mathematical step. This approach bridges the gap between theoretical concepts and practical implementation.

Contents

1 Introduction to Mathematical Reasoning in ML

1.1 The Importance of Understanding Operations

Machine learning algorithms are built on mathematical foundations where each operation serves a specific purpose. Understanding these operations is crucial because:

- It enables proper algorithm selection for specific problems
- It helps diagnose and fix model performance issues

- It facilitates custom algorithm modifications
- It provides intuition for hyperparameter tuning
- It supports informed feature engineering decisions

2 Linear Regression: Step-by-Step Mathematical Journey

2.1 Problem Setup

Let's predict house prices based on size (sq ft) and number of bedrooms.

Training Data:

Size (sq ft)	Bedrooms	Price ($1000)
1500	3	300
2000	4	450
1200	2	250
1800	3	400

2.2 Matrix Representation and Operations

2.2.1 Why We Use Matrix Notation

Matrix notation allows us to handle multiple features and data points efficiently. It enables vectorized computations that are computationally efficient and mathematically elegant.

Feature Matrix X and Target Vector y:

$$X = \begin{bmatrix} 1 & 1500 & 3 \\ 1 & 2000 & 4 \\ 1 & 1200 & 2 \\ 1 & 1800 & 3 \end{bmatrix}, \quad y = \begin{bmatrix} 300 \\ 450 \\ 250 \\ 400 \end{bmatrix}$$

Why the first column is all 1s: This represents the intercept term (bias). Without it, our model would always pass through the origin, which is rarely appropriate for real-world data.

2.2.2 Operation 1: Compute $X^T X$

Why we do this: $X^T X$ creates the covariance matrix that captures how features relate to each other. This is essential for understanding the relationships between different features.

How we do it:

$$X^T = \begin{bmatrix} 1 & 1 & 1 & 1 \\ 1500 & 2000 & 1200 & 1800 \\ 3 & 4 & 2 & 3 \end{bmatrix}$$

$$X^T X = \begin{bmatrix} 1 & 1 & 1 & 1 \\ 1500 & 2000 & 1200 & 1800 \\ 3 & 4 & 2 & 3 \end{bmatrix} \begin{bmatrix} 1 & 1500 & 3 \\ 1 & 2000 & 4 \\ 1 & 1200 & 2 \\ 1 & 1800 & 3 \end{bmatrix}$$

Let's compute step by step:

Element (1,1): $1 \times 1 + 1 \times 1 + 1 \times 1 + 1 \times 1 = 4$
Element (1,2): $1 \times 1500 + 1 \times 2000 + 1 \times 1200 + 1 \times 1800 = 6500$
Element (1,3): $1 \times 3 + 1 \times 4 + 1 \times 2 + 1 \times 3 = 12$
Element (2,2): $1500 \times 1500 + 2000 \times 2000 + 1200 \times 1200 + 1800 \times 1800 = 10,930,000$
Element (2,3): $1500 \times 3 + 2000 \times 4 + 1200 \times 2 + 1800 \times 3 = 20,300$
Element (3,3): $3 \times 3 + 4 \times 4 + 2 \times 2 + 3 \times 3 = 38$

Due to symmetry:

$$X^T X = \begin{bmatrix} 4 & 6500 & 12 \\ 6500 & 10,930,000 & 20,300 \\ 12 & 20,300 & 38 \end{bmatrix}$$

What this tells us: The diagonal elements show the sum of squares for each feature, while off-diagonal elements show how features co-vary.

2.2.3 Operation 2: Compute $X^T y$

Why we do this: $X^T y$ measures how each feature correlates with the target variable. This tells us which features are most strongly associated with house prices.

How we do it:

$$X^T y = \begin{bmatrix} 1 & 1 & 1 & 1 \\ 1500 & 2000 & 1200 & 1800 \\ 3 & 4 & 2 & 3 \end{bmatrix} \begin{bmatrix} 300 \\ 450 \\ 250 \\ 400 \end{bmatrix} = \begin{bmatrix} 300 + 450 + 250 + 400 \\ 1500 \times 300 + 2000 \times 450 + 1200 \times 250 + 1800 \times 400 \\ 3 \times 300 + 4 \times 450 + 2 \times 250 + 3 \times 400 \end{bmatrix} =$$

2.2.4 Operation 3: Compute $(X^T X)^{-1}$

Why we do this: The inverse of $X^T X$ is needed to solve the normal equations. It represents the precision matrix, which shows conditional independence relationships between features.

How we do it (conceptually): For a 3×3 matrix, we would use methods like Gaussian elimination or LU decomposition. The actual computation is complex, but let's denote the result as:

$$(X^T X)^{-1} = \begin{bmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{bmatrix}$$

Why inversion is tricky: If $X^T X$ is singular (not invertible), it means our features are linearly dependent. This is why we often use regularization in practice.

2.2.5 Operation 4: Compute $w = (X^T X)^{-1} X^T y$

Why we do this: This gives us the optimal weights that minimize the sum of squared errors between predicted and actual prices.

How it works: Multiplying the inverse covariance matrix by the feature-target correlations gives us the weights that best explain the relationship.

If we had the actual inverse, we would compute:

$$w = \begin{bmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{bmatrix} \begin{bmatrix} 1400 \\ 2,430,000 \\ 4700 \end{bmatrix} = \begin{bmatrix} w_0 \\ w_1 \\ w_2 \end{bmatrix}$$

2.3 Numerical Example with Simplified Data

Let's use a simpler example to see actual numbers:

Data: $X = \begin{bmatrix} 1 & 1 \\ 1 & 2 \\ 1 & 3 \end{bmatrix}, y = \begin{bmatrix} 1 \\ 2 \\ 3 \end{bmatrix}$

Step 1: $X^T X = \begin{bmatrix} 1 & 1 & 1 \\ 1 & 2 & 3 \end{bmatrix} \begin{bmatrix} 1 & 1 \\ 1 & 2 \\ 1 & 3 \end{bmatrix} = \begin{bmatrix} 3 & 6 \\ 6 & 14 \end{bmatrix}$

Step 2: $X^T y = \begin{bmatrix} 1 & 1 & 1 \\ 1 & 2 & 3 \end{bmatrix} \begin{bmatrix} 1 \\ 2 \\ 3 \end{bmatrix} = \begin{bmatrix} 6 \\ 14 \end{bmatrix}$

Step 3: $(X^T X)^{-1} = \frac{1}{3 \times 14 - 6 \times 6} \begin{bmatrix} 14 & -6 \\ -6 & 3 \end{bmatrix} = \frac{1}{6} \begin{bmatrix} 14 & -6 \\ -6 & 3 \end{bmatrix}$

Step 4: $w = \frac{1}{6} \begin{bmatrix} 14 & -6 \\ -6 & 3 \end{bmatrix} \begin{bmatrix} 6 \\ 14 \end{bmatrix} = \frac{1}{6} \begin{bmatrix} 14 \times 6 + (-6) \times 14 \\ -6 \times 6 + 3 \times 14 \end{bmatrix} = \frac{1}{6} \begin{bmatrix} 0 \\ 6 \end{bmatrix} = \begin{bmatrix} 0 \\ 1 \end{bmatrix}$

Final model: $\hat{y} = 0 + 1 \times x$, which perfectly fits our data!

3 Decision Trees: The Mathematics of Splitting

3.1 Problem Setup

Let's classify whether a customer will buy a product based on age and income.

Training Data:

Age	Income ($1000)	Student	Buys
Youth	High	No	No
Youth	High	Yes	Yes
Middle	High	No	Yes
Senior	Medium	No	Yes
Senior	Low	Yes	Yes
Senior	Low	No	No

3.2 Information Gain Calculation

3.2.1 Why We Use Information Theory

Information gain measures how much a split reduces uncertainty about the target variable. We want splits that create pure child nodes.

3.2.2 Step 1: Calculate Parent Impurity

Before any split: - Total instances: 6 - "Yes" class: 4 instances - "No" class: 2 instances

Using Gini Impurity:

$$I_{\text{parent}} = 1 - \left(\frac{4}{6}\right)^2 - \left(\frac{2}{6}\right)^2 = 1 - \frac{16}{36} - \frac{4}{36} = 1 - \frac{20}{36} = \frac{16}{36} \approx 0.444$$

Using Entropy:

$$H_{\text{parent}} = -\frac{4}{6}\log_2\frac{4}{6} - \frac{2}{6}\log_2\frac{2}{6} \approx 0.918$$

3.2.3 Step 2: Evaluate Split on "Age"

Why we consider different splits: We want to find the feature that best separates the classes.

Age = Youth: - Instances: 2 - Yes: 1, No: 1 - Gini: $1 - (0.5)^2 - (0.5)^2 = 0.5$

Age = Middle: - Instances: 1 - Yes: 1, No: 0 - Gini: $1 - (1)^2 - (0)^2 = 0$

Age = Senior: - Instances: 3 - Yes: 2, No: 1 - Gini: $1 - (\frac{2}{3})^2 - (\frac{1}{3})^2 = 1 - \frac{4}{9} - \frac{1}{9} = \frac{4}{9} \approx 0.444$

Weighted average impurity:

$$I_{\text{age}} = \frac{2}{6} \times 0.5 + \frac{1}{6} \times 0 + \frac{3}{6} \times 0.444 \approx 0.389$$

Information Gain:

$$IG_{\text{age}} = I_{\text{parent}} - I_{\text{age}} = 0.444 - 0.389 = 0.055$$

3.2.4 Step 3: Evaluate Split on "Student"

Student = Yes: - Instances: 2 - Yes: 2, No: 0 - Gini: $1 - (1)^2 - (0)^2 = 0$

Student = No: - Instances: 4 - Yes: 2, No: 2 - Gini: $1 - (0.5)^2 - (0.5)^2 = 0.5$

Weighted average impurity:

$$I_{\text{student}} = \frac{2}{6} \times 0 + \frac{4}{6} \times 0.5 \approx 0.333$$

Information Gain:

$$IG_{\text{student}} = 0.444 - 0.333 = 0.111$$

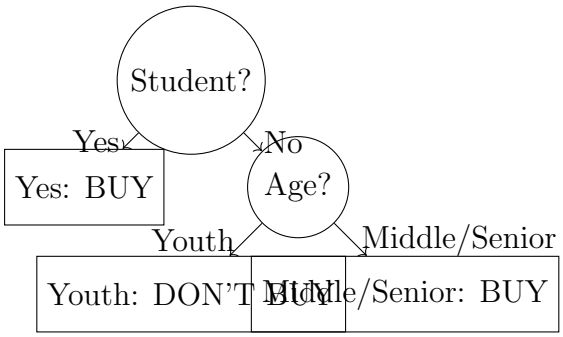
3.2.5 Step 4: Compare and Choose Best Split

Why we choose the highest information gain: This gives us the split that most reduces uncertainty.

Feature	Information Gain
Age	0.055
Student	0.111
Income	(calculate similarly)

We choose "Student" as the root split because it gives the highest information gain.

3.3 Visualizing the Decision Process



4 Support Vector Machines: Maximum Margin Geometry

4.1 Problem Setup

Binary classification with 2D data for visualization.

Training Data:

X1	X2	Class
1	2	+1
2	3	+1
2	1	-1
3	2	-1

4.2 Mathematical Formulation

4.2.1 Why Maximum Margin?

The margin represents the "safety zone" around the decision boundary. A larger margin means better generalization to new data.

4.2.2 Primal Problem

We want to find w and b such that:

$$\begin{aligned} \min_{w,b} \quad & \frac{1}{2}\|w\|^2 \\ \text{subject to} \quad & y_i(w^T x_i + b) \geq 1 \quad \forall i \end{aligned}$$

Why we minimize $\frac{1}{2}\|w\|^2$: The margin width is $\frac{2}{\|w\|}$, so minimizing $\|w\|$ maximizes the margin.

Why constraints are $y_i(w^T x_i + b) \geq 1$: This ensures all points are at least distance 1 from the decision boundary.

4.2.3 Lagrangian Dual Formulation

Why we use the dual: The dual problem reveals the "support vectors" and enables the kernel trick for non-linear classification.

The Lagrangian:

$$L(w, b, \alpha) = \frac{1}{2}\|w\|^2 - \sum_{i=1}^{n} \alpha_i [y_i(w^T x_i + b) - 1]$$

Taking derivatives and setting to zero:

$$\frac{\partial L}{\partial w} = w - \sum_{i=1}^{n} \alpha_i y_i x_i = 0 \Rightarrow w = \sum_{i=1}^{n} \alpha_i y_i x_i$$

$$\frac{\partial L}{\partial b} = -\sum_{i=1}^{n} \alpha_i y_i = 0 \Rightarrow \sum_{i=1}^{n} \alpha_i y_i = 0$$

Substituting back, we get the dual problem:

$$\max_{\alpha} \sum_{i=1}^{n} \alpha_i - \frac{1}{2} \sum_{i=1}^{n} \sum_{j=1}^{n} \alpha_i \alpha_j y_i y_j x_i^T x_j$$

subject to $\alpha_i \geq 0$ and $\sum_{i=1}^{n} \alpha_i y_i = 0$.

4.2.4 Numerical Example

Let's solve our small problem. The positive class points are (1,2) and (2,3), negative class points are (2,1) and (3,2).

The dual objective becomes:

$$\max_{\alpha} \alpha_1 + \alpha_2 + \alpha_3 + \alpha_4 - \frac{1}{2}[\alpha_1^2(1^2 + 2^2) + \alpha_2^2(2^2 + 3^2) + \cdots + 2\alpha_1\alpha_2 y_1 y_2 (1 \times 2 + 2 \times 3) + \cdots]$$

After solving (typically using quadratic programming), we find that only the points closest to the boundary have non-zero α values - these are our support vectors.

4.3 Kernel Trick Demonstration

Why kernels are needed: When data is not linearly separable, we can map it to a higher-dimensional space where it becomes separable.

Example: Consider the XOR problem:

X1	X2	Class
0	0	-1
0	1	+1
1	0	+1
1	1	-1

This is not linearly separable in 2D. But if we map to 3D using $\phi(x) = (x_1, x_2, x_1 x_2)$, it becomes separable!

The kernel $K(x_i, x_j) = \phi(x_i)^T \phi(x_j)$ lets us compute this without explicitly computing $\phi(x)$.

For polynomial kernel of degree 2:

$$K(x_i, x_j) = (x_i^T x_j + 1)^2$$

This corresponds to implicit mapping to a higher-dimensional space without the computational cost.

5 k-Nearest Neighbors: Distance and Voting

5.1 Problem Setup

Classify a new customer based on similar existing customers.

Training Data:

Age	Income	Credit Score	Good Customer
25	40	650	No
35	60	700	Yes
45	80	750	Yes
30	50	600	No
40	70	720	Yes

New customer: Age=38, Income=65, Credit Score=710, k=3

5.2 Distance Calculations

5.2.1 Why Distance Matters

Distance measures similarity. Closer points are more similar and should have more influence on the prediction.

5.2.2 Euclidean Distance Calculation

For the new point (38,65,710) and training point (35,60,700):

$$d = \sqrt{(38-35)^2 + (65-60)^2 + (710-700)^2} = \sqrt{9+25+100} = \sqrt{134} \approx 11.58$$

Compute distances to all points:

Point	Distance	Class
(25,40,650)	$\sqrt{(38-25)^2 + (65-40)^2 + (710-650)^2} \approx 65.19$	No
(35,60,700)	$\sqrt{134} \approx 11.58$	Yes
(45,80,750)	$\sqrt{(38-45)^2 + (65-80)^2 + (710-750)^2} \approx 43.57$	Yes
(30,50,600)	$\sqrt{(38-30)^2 + (65-50)^2 + (710-600)^2} \approx 111.36$	No
(40,70,720)	$\sqrt{(38-40)^2 + (65-70)^2 + (710-720)^2} \approx 11.36$	Yes

5.2.3 Nearest Neighbors and Voting

Sort by distance and pick top k=3:

1. (40,70,720): distance=11.36, class=Yes
2. (35,60,700): distance=11.58, class=Yes
3. (45,80,750): distance=43.57, class=Yes

Voting: All 3 neighbors say "Yes", so we predict "Good Customer = Yes"

5.2.4 Weighted k-NN

Why weighting is useful: Closer points should have more influence than farther points.

Using inverse distance weighting:

$$\text{Weight} = \frac{1}{\text{distance}}$$

Weighted votes:

- Point 1: weight = 1/11.36 0.088, vote = Yes
- Point 2: weight = 1/11.58 0.086, vote = Yes
- Point 3: weight = 1/43.57 0.023, vote = Yes

Total weighted vote for Yes = 0.197, still clearly Yes.

6 K-Means Clustering: Iterative Centroid Updates

6.1 Problem Setup

Cluster customers based on spending and frequency.

Customer Data:

Spending ($100)	Visits/Month
2	8
4	6
3	5
8	2
9	3
7	1

We want k=2 clusters.

6.2 Algorithm Steps

6.2.1 Step 1: Initialization

Why random initialization: We need starting points for our clusters. K-means++ provides better initialization.

Let's randomly choose: - Cluster 1 center: (2,8) - Cluster 2 center: (8,2)

6.2.2 Step 2: Assignment

Why assignment step: Each point should belong to the closest cluster.

Compute distances to both centers:

For point (4,6):

$$d_1 = \sqrt{(4-2)^2 + (6-8)^2} = \sqrt{4+4} = \sqrt{8} \approx 2.83$$
$$d_2 = \sqrt{(4-8)^2 + (6-2)^2} = \sqrt{16+16} = \sqrt{32} \approx 5.66$$

Assign to cluster 1 (closer).

Repeat for all points:

Point	d1	d2	Assignment
(2,8)	0	$\sqrt{72} \approx 8.49$	Cluster 1
(4,6)	2.83	5.66	Cluster 1
(3,5)	3.16	5.83	Cluster 1
(8,2)	8.49	0	Cluster 2
(9,3)	9.22	1.41	Cluster 2
(7,1)	8.60	2.24	Cluster 2

6.2.3 Step 3: Centroid Update

Why update centroids: The centers should be at the mean of their assigned points for optimal cluster representation.

Cluster 1 points: (2,8), (4,6), (3,5)

$$\text{New center}_1 = \left(\frac{2+4+3}{3}, \frac{8+6+5}{3}\right) = (3, 6.33)$$

Cluster 2 points: (8,2), (9,3), (7,1)

$$\text{New center}_2 = \left(\frac{8+9+7}{3}, \frac{2+3+1}{3}\right) = (8, 2)$$

6.2.4 Step 4: Repeat Until Convergence

We repeat assignment and update until centers stop moving significantly.

After convergence, we might get: - Cluster 1 (low spending, high frequency): center around (3,6) - Cluster 2 (high spending, low frequency): center around (8,2)

6.3 Mathematical Objective

The K-means objective:

$$\min \sum_{k=1}^{K} \sum_{x_i \in C_k} \|x_i - \mu_k\|^2$$

Why squared Euclidean distance: It emphasizes larger errors and has nice mathematical properties (differentiability).

At each iteration, we're guaranteed to decrease this objective until we reach a local minimum.

7 Neural Networks: Forward and Backward Propagation

7.1 Problem Setup

Simple binary classification with 2D data.

Training Data:

X1	X2	Class
0	0	0
0	1	1
1	0	1
1	1	0

This is the XOR problem - not linearly separable!

7.2 Network Architecture

We'll use a simple network: - Input layer: 2 neurons (X1, X2) - Hidden layer: 2 neurons with sigmoid activation - Output layer: 1 neuron with sigmoid activation

7.3 Forward Propagation Example

Let's use specific weights and biases for demonstration:

Weights and Biases: - $W^{[1]} = \begin{bmatrix} 2 & 2 \\ 2 & 2 \end{bmatrix}$, $b^{[1]} = \begin{bmatrix} -1 \\ -3 \end{bmatrix}$ - $W^{[2]} = \begin{bmatrix} 2 & -4 \end{bmatrix}$, $b^{[2]} = \begin{bmatrix} -1 \end{bmatrix}$

Forward pass for input (0,1):

Layer 1:

$$z^{[1]} = W^{[1]}x + b^{[1]} = \begin{bmatrix} 2 & 2 \\ 2 & 2 \end{bmatrix} \begin{bmatrix} 0 \\ 1 \end{bmatrix} + \begin{bmatrix} -1 \\ -3 \end{bmatrix} = \begin{bmatrix} 2 \times 0 + 2 \times 1 - 1 \\ 2 \times 0 + 2 \times 1 - 3 \end{bmatrix} = \begin{bmatrix} 1 \\ -1 \end{bmatrix}$$

$$a^{[1]} = \sigma(z^{[1]}) = \begin{bmatrix} \frac{1}{1+e^{-1}} \\ \frac{1}{1+e^{1}} \end{bmatrix} \approx \begin{bmatrix} 0.731 \\ 0.269 \end{bmatrix}$$

Layer 2:

$$z^{[2]} = W^{[2]}a^{[1]} + b^{[2]} = \begin{bmatrix} 2 & -4 \end{bmatrix} \begin{bmatrix} 0.731 \\ 0.269 \end{bmatrix} + \begin{bmatrix} -1 \end{bmatrix} = 2 \times 0.731 + (-4) \times 0.269 - 1 \approx -0.854$$

$$a^{[2]} = \sigma(z^{[2]}) = \frac{1}{1 + e^{0.854}} \approx 0.299$$

Prediction: 0.299 (close to actual class 1, but not perfect)

7.4 Backward Propagation Example

Loss calculation (binary cross-entropy):

$$L = -[y \log(a^{[2]}) + (1 - y) \log(1 - a^{[2]})] = -[1 \times \log(0.299) + 0 \times \log(0.701)] \approx 1.206$$

Backward pass:

Output layer gradients:

$$\frac{\partial L}{\partial a^{[2]}} = -\frac{y}{a^{[2]}} + \frac{1 - y}{1 - a^{[2]}} = -\frac{1}{0.299} + \frac{0}{0.701} \approx -3.344$$

$$\frac{\partial a^{[2]}}{\partial z^{[2]}} = a^{[2]}(1 - a^{[2]}) = 0.299 \times 0.701 \approx 0.209$$

$$\delta^{[2]} = \frac{\partial L}{\partial z^{[2]}} = \frac{\partial L}{\partial a^{[2]}} \cdot \frac{\partial a^{[2]}}{\partial z^{[2]}} = -3.344 \times 0.209 \approx -0.700$$

Hidden layer gradients:

$$\frac{\partial L}{\partial W^{[2]}} = \delta^{[2]} a^{[1]T} = -0.700 \times \begin{bmatrix} 0.731 \\ 0.269 \end{bmatrix}^T = \begin{bmatrix} -0.512 & -0.188 \end{bmatrix}$$

$$\frac{\partial L}{\partial b^{[2]}} = \delta^{[2]} = -0.700$$

$$\delta^{[1]} = (W^{[2]T} \delta^{[2]}) \odot \sigma'(z^{[1]}) = \begin{bmatrix} 2 \\ -4 \end{bmatrix} \times -0.700 \odot \begin{bmatrix} 0.731 \times (1 - 0.731) \\ 0.269 \times (1 - 0.269) \end{bmatrix} \approx \begin{bmatrix} -0.280 \\ -0.543 \end{bmatrix}$$

7.5 Weight Update

Using learning rate $\eta = 0.1$:

$$W^{[2]}_{\text{new}} = W^{[2]} - \eta \frac{\partial L}{\partial W^{[2]}} = \begin{bmatrix} 2 & -4 \end{bmatrix} - 0.1 \times \begin{bmatrix} -0.512 & -0.188 \end{bmatrix} = \begin{bmatrix} 2.051 & -3.981 \end{bmatrix}$$

$$b^{[2]}_{\text{new}} = b^{[2]} - \eta \frac{\partial L}{\partial b^{[2]}} = -1 - 0.1 \times (-0.700) = -0.930$$

Similarly update $W^{[1]}$ and $b^{[1]}$.

7.6 Why This Works

- **Forward propagation:** Computes predictions using current weights
- **Loss calculation:** Measures how wrong our predictions are
- **Backward propagation:** Uses chain rule to compute how much each weight contributes to the error
- **Weight update:** Adjusts weights to reduce future errors

After many iterations, the network learns to approximate the XOR function!

8 Conclusion

8.1 Key Mathematical Insights

- **Linear Regression:** Matrix operations efficiently solve the normal equations for optimal linear fit
- **Decision Trees:** Information theory guides optimal splits for pure classification
- **SVMs:** Optimization theory finds maximum margin hyperplanes for robust classification
- **k-NN:** Distance metrics and voting provide simple but powerful instance-based learning
- **K-Means:** Iterative centroid updates minimize within-cluster variance
- **Neural Networks:** Gradient descent and chain rule enable learning complex non-linear functions

8.2 The Importance of Understanding Operations

Understanding the "why" and "how" behind mathematical operations in machine learning:

- Enables informed algorithm selection based on problem characteristics
- Facilitates debugging and performance improvement
- Supports custom modifications for specific applications
- Provides intuition for hyperparameter tuning
- Builds foundation for developing new algorithms

Each mathematical operation serves a specific purpose in the learning process, and understanding this relationship is key to mastering machine learning.