ISSN No:-2456-2165 https://doi.org/10.38124/ijisrt/25sep829

Linear Algebra as a Mathematical Foundation for Machine Learning Algorithms

Mubarak Sadudeen[1]

[1] Wellspring College, Ikeja, Lagos, Nigeria

Publication Date: 2025/10/03

Abstract: Machine Learning (ML) is no longer a novel academic domain or even a niche within a specific technology but has become a technological engine that transforms many sectors of life: healthcare and finance, transportation and entertainment, etc. The effectiveness of such algorithms in analysing complex data, recognising patterns and giving precise predictions is frequently viewed as a kind of computational alchemy. This impression is however, false and does not represent a hard mathematical framework which is fundamental to both the knowledge and development of the field. Linear algebra forms the basis of this foundation by a very huge margin. In this paper, a detailed assessment will be given explaining the inseparable nature of linear algebra as the language of machine learning. The goal is to systematically break down the important ML algorithms, including both simplistic linear regression, but also more complicated deep learning models, and explicitly trace their basic mechanisms to the underlying linear algebraic operations, including: matrix multiplication, transformations of vectors spaces, and manipulations of tensors. This review will help to dispel the mystique of the black box nature of ML by showing that data representation, model operation, and optimization are all intrinsically linear algebraic operations. This synthesis is valuable to students in need of a more conceptual grasp, researchers in need of creating new algorithms, and practitioners in need of debugging, optimizing, and innovating their ML pipelines. Detailed understanding of these mathematical foundations is not only academic but a requirement to be able to master the practice and innovate in the sphere of artificial intelligence.

Keywords: *Linear Algebra, Machine Learning, Matrices, Vectors, Dimensionality Reduction, Optimization, Neural Networks, Tensors.*

How to Cite: Mubarak Sadudeen (2025) Linear Algebra as a Mathematical Foundation for Machine Learning Algorithms. *International Journal of Innovative Science and Research Technology*, 10(9), 2257-2262. https://doi.org/10.38124/ijisrt/25sep829

I. INTRODUCTION

Machine learning, a sub-field of artificial intelligence is officially described as the research of those computer algorithms that may be enhanced by experience and by means of data (Mitchell, 1997). It is broadly divided into supervised learning, where the models are trained to map input data to known output labels; unsupervised learning, where the model is trained to learn the patterns based on the input data without the responses being provided; and reinforcement learning, as an agent is trained to make decisions by taking action and being rewarded in an environment (Sutton and Barto, 2018). The widespread use of ML applications has developed a tools-centric culture in which sophisticated models may be written in only a few lines of high-level code and the mathematical engines behind them may often be hidden. This abstraction results in the major issue, that is, practitioners can effectively use models without being able to sense their failure, maximize their performance out of the baseline parameters or advance on their architectures because of not knowing the underlying principles.

The functioning and effectiveness of these algorithms are not a secret, but are deeply rooted in serious mathematics, especially linear algebra. Linear algebra gives us the vocabulary and syntax of explaining and performing the computations that underlie ML. It is the system that data is organized on, models are developed, and learning is operationalized. As an example, a single data element, i.e. the health statistics of a patient or the features of a product is inherently modeled as a high-dimensional vector. A full set of such data, such as thousands or millions of such data points, is organized in a matrix. This basic process of making a prediction in a linear model is simplified to a dot product of a weight vector and a feature vector. Moreover, the learning process, or fitting the model parameters to the minimal error, is a type of optimization problem, which is usually addressed with methods based on the matrix calculus, including gradient descent (Bishop, 2006).

The underlying connection of linear algebra to machine learning is long-standing in textbook literature. ML as a story of the linear algebraic origin of the former can be explicitly constructed in textbooks like Strang's "Linear Algebra and Learning from Data" (2019) and Goodfellow's "Deep Learning" (2016). This body of work establishes the

ISSN No:-2456-2165 https://doi.org/10.38124/ijisrt/25sep829

fact that linear algebra is not merely a helpful tool but the very medium of computational learning. In order to describe this relationship in a systematic way, it is informative to directly map major concepts of linear algebra to their key uses in machine learning paradigms, as in Table 1.

Table 1 Correspondence Between Key Linear Algebra Concepts and Machine Learning Applications

Linear Algebra Concept	Definition & Mathematical Significance	Primary ML Application(s)	Impact on ML Algorithm
Vectors & Matrices	Vectors are ordered lists of numbers (scalars) representing points or directions in space. Matrices are 2D arrays of numbers representing linear transformations or datasets.	Universal data representation. All ML models.	Provides the structure for input data (design matrix), model parameters (weight matrix), and output predictions.
Systems of Linear Equations	A set of equations with multiple variables that can be expressed in matrix form as **A**x = **b**.	Linear Regression, Least Squares Optimization.	The core problem of fitting a linear model is solving a system of equations to find the optimal weights.
Eigen-Decomposition	Factorizing a square matrix into its eigenvectors and eigenvalues, which reveal the matrix's fundamental properties and transformations.	Principal Component Analysis (PCA), Spectral Clustering.	Used for dimensionality reduction by identifying the directions (eigenvectors) of maximum variance in the data.
Singular Value Decomposition (SVD)	Factorizing any rectangular matrix into singular vectors and singular values, generalizing eigen-decomposition.	Recommender Systems, Latent Semantic Analysis, Matrix Completion.	Uncovers latent features in data for tasks like collaborative filtering and low-rank approximations.
Matrix Calculus & Gradients	The extension of calculus to matrices and vectors, allowing for the computation of derivatives of functions with matrix inputs/outputs.	Training Neural Networks, Gradient-Based Optimization (e.g., SGD, Adam).	Enables the calculation of the error gradient, which is essential for updating model parameters during learning.
Norms	Functions that assign a strictly positive length or size to a vector (e.g., L2 norm for distance, L1 norm for sparsity).	Regularization (L1/Lasso, L2/Ridge), Loss Functions.	Used to constrain model complexity and prevent overfitting by adding a penalty to the loss function.
Tensors	Multi-dimensional arrays generalizing vectors (1D) and matrices (2D) to higher dimensions (3D, 4D, etc.).	Deep Learning (Convolutional Neural Networks, Transformers).	Efficiently represents complex data like images (3D: height, width, channels) and batches of data (4D: batch, height, width, channels).

Source: Adapted from Strang (2019) and Goodfellow et al. (2016)

Therefore, the primary goal of this paper is to demystify machine learning by systematically exploring the linear algebra at its core. This review will dissect how algorithms are constructed from first principles of vectors, matrices, and their operations, moving from basic models to advanced architectures. This approach provides a unifying lens through which the entire field of machine learning can be understood not as a collection of disparate tricks, but as a coherent application of linear mathematical principles.

II. RESULTS

➢ *An Overview of Foundational Linear Algebra Concepts*

The journey into the mathematical heart of machine learning begins with a firm grasp of the basic elements of linear algebra. These elements are not abstract mathematical curiosities but are the direct building blocks of every data structure and computation in ML. A scalar is a single number, a singular entity representing magnitude alone, such as a single temperature reading or a model's learning rate. A vector is an ordered array of numbers, a one-dimensional list that represents a point in a multi-dimensional space. For example, the features of a single house—its square footage, number of bedrooms, and age—can be represented as a vector [2100, 4, 20], placing it within a 3-dimensional feature space. This conceptualization is fundamental to all of ML, as it allows for the geometric interpretation of data (Saxe, 2021).

A matrix is a two-dimensional array of numbers, a rectangular grid that can represent an entire dataset. Each row of a matrix typically corresponds to a single data point (a vector), and each column corresponds to a specific feature. This structure, known as the design matrix, is the primary format for input data in most ML algorithms. A tensor is the generalization of these concepts to higher dimensions. While a vector is a first-order tensor and a matrix is a second-order tensor, a third-order tensor could represent a batch of images, and a fourth-order tensor could represent a batch of multi-channel images. The operations defined on these structures are equally critical. Vector addition and scalar multiplication allow for the translation and scaling of data points. The dot product (or inner product) of two vectors yields a scalar that measures their similarity and is the fundamental operation behind calculating predictions and similarities (Petersen & Pedersen, 2012).

ISSN No:-2456-2165 https://doi.org/10.38124/ijisrt/25sep829

Beyond these basic operations, key concepts define the behavior of these mathematical objects. A linear transformation is a function that maps vectors from one space to another while preserving the operations of vector addition and scalar multiplication. These transformations are represented by matrices; multiplying a vector by a matrix applies the transformation to the vector. The rank of a matrix reveals the dimensionality of the vector space spanned by its columns, indicating the number of linearly independent features in a dataset. A low-rank matrix suggests redundant or correlated features. The determinant is a scalar value that provides information about the scaling factor of the linear transformation described by the matrix and whether that transformation is orientation-preserving. Perhaps most crucially, the inverse of a matrix, when it exists, allows for solving systems of linear equations. Understanding these concepts is not a mere academic exercise; it is essential for comprehending how models manipulate data, why certain computations succeed or fail (e.g., a non-invertible matrix halting a regression calculation), and how the geometric properties of data impact learning (Trefethen & Bau, 1997). A visual representation of these operations, such as Figure 1, can greatly aid in building intuition.

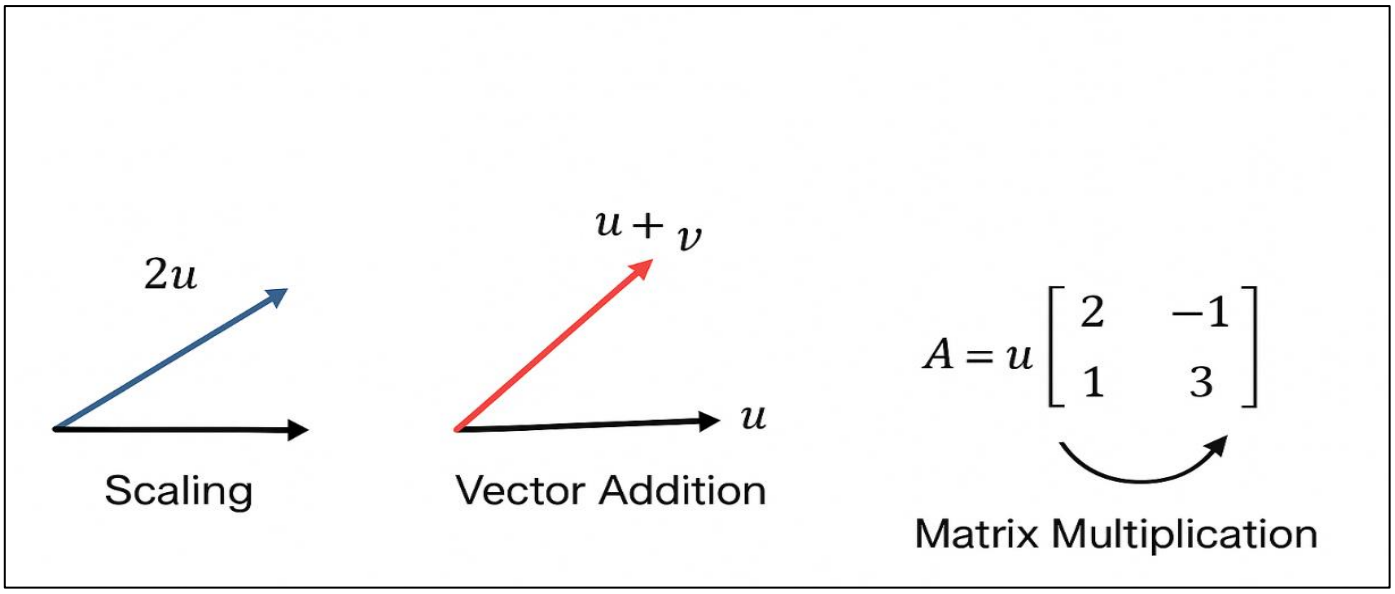

Fig 1 Geometric Interpretation of Basic Linear Algebra Operations

➢ *Linear Algebra in Model Representation and Operation*

- *Vectors and Matrices for Data Representation*

The first step in any machine learning process is how data is presented, and this is closely related to linear algebra. A raw data set, such as a CSV file with house details, is naturally organized in a matrix format. Each row represents a specific house, which is an individual data point. Each column represents a different characteristic of those houses, like square footage, zip code, or year built. When this data is loaded, it is converted into a design matrix, usually called X. This matrix has m rows and n columns, where m is the number of data points and n is the number of features (Géron, 2022). This matrix format is efficient for computation and allows batch processing with optimized linear algebra libraries.

A basic predictive model, like linear regression, uses matrix and vector operations to make predictions. The goal is to predict a target value, such as house price, based on a set of feature values. This is done by calculating a weighted sum of the input features, which is essentially the dot product between a weight vector and the feature vector, plus a bias term. For a single prediction, this is represented as $\hat{y} = w \cdot x + b$. For the entire dataset, this operation is expressed as a matrix multiplication: $\hat{y} = Xw + b$, where X is the design matrix, w is a column vector of weights, and $\hat{y}$ is a column vector of predictions. This notation is not only elegant but also allows modern hardware, like GPUs, to perform millions of calculations quickly and efficiently, forming the foundation of training and prediction (VanderPlas, 2023).

- *Solving Systems of Equations for Regression*

Training a linear regression model involves finding the best weight vector w that minimizes the difference between predicted and actual values.

The most common method, Ordinary Least Squares (OLS), uses a sum-of-squares loss function to measure this difference. Minimizing this loss leads to a system of equations known as the normal equations: (X^T X) w = X^T y (Deisenroth, Faisal, & Ong, 2020). Here, X^T is the transpose of the design matrix. The optimal weights, w*, are found by solving w* = (X^T X)^{-1} X^T y.

This process shows how a fundamental machine learning task translates into a fundamental linear algebra problem: solving a system of equations.

The matrix (X^T X) must be inverted, highlighting the importance of concepts like matrix invertibility and rank. If the features in X are dependent (i.e., X^T X is singular), the solution becomes unstable or impossible. For large datasets or to ensure numerical stability, direct inversion is avoided.

ISSN No:-2456-2165 https://doi.org/10.38124/ijisrt/25sep829

Instead, methods like QR decomposition or Singular Value Decomposition (SVD) are used to solve the system without inverting X^T X directly (Trefethen & Bau, 1997). Therefore, the entire process of linear regression, from setup to solution, is an application of linear algebra.

- *Eigen-Decomposition in Dimensionality Reduction*

A common issue in machine learning is the curse of dimensionality, where a large number of features can make models inefficient, prone to overfitting, and hard to interpret.

Dimensionality reduction techniques address this by projecting data into a lower-dimensional space while keeping its main structure. Principal Component Analysis (PCA) is a leading method for linear dimensionality reduction and is based on eigen-decomposition (Jolliffe & Cadima, 2016).

The process starts with calculating the covariance matrix of the centered data. This matrix, which is square and symmetric, shows the variance of each feature on its diagonal and the covariance between pairs of features off-diagonal. PCA finds the directions in the feature space where data varies the most. These directions are the eigenvectors of the covariance matrix. The corresponding eigenvalues show how much variance is captured by each eigenvector. The eigenvectors with the largest eigenvalues are the principal components. The data is then projected onto the subspace formed by the top-k principal components, resulting in a lower-dimensional representation that keeps most of the original data's variability. This entire process is a clear example of how matrix properties, such as eigenvectors and eigenvalues, reveal the underlying structure of high-dimensional data.

Table 2 The PCA Algorithm as a Linear Algebraic Procedure

Step	Description	Linear Algebra Operation
1.	Center the data	Subtract the mean vector from each row of the data matrix **X** to get **X'**.
2.	Compute Covariance	Calculate the covariance matrix **Σ** = (1/(m-1)) * X'^T X'.
3.	Eigen-decomposition	Factorize the covariance matrix: **Σ** = **QΛQ^T**, where **Λ** is a diagonal matrix of eigenvalues and **Q** is a matrix of eigenvectors.
4.	Select Components	Sort eigenvectors in **Q** by descending eigenvalues. Select the top-k columns of **Q** to form matrix **W** (the projection matrix).
5.	Project Data	Transform the original data to the new subspace: **Z** = **X' W**. **Z** is the new m x k low-dimensional dataset.

Source: Based on Jolliffe & Cadima (2016)

➢ *Linear Algebra in Optimization and Advanced Models*

- *Matrix Calculus and Gradient-Based Optimization*

For models more complex than linear regression, an analytical solution like the normal equations is not feasible. The training of neural networks and other sophisticated models relies on iterative optimization algorithms, the most famous of which is gradient descent. The generalization of calculus to vectors and matrices is essential for this process. The gradient of a scalar-valued function (like a loss function J(w)) with respect to a vector of parameters **w** is itself a vector. Denoted ∇_w J(w), this gradient points in the direction of the steepest ascent of the function (Saxe, 2021).

The gradient descent update rule, w = w - α ∇_w J(w), is a vector subtraction operation. The learning rate α is a scalar that modulates the size of the step taken in the direction of the negative gradient. In modern deep learning, frameworks like TensorFlow and PyTorch use automatic differentiation to compute these gradients efficiently for incredibly complex functions comprising millions of parameters. This process, known as backpropagation, is essentially a repeated application of the chain rule from calculus through the computation graph of the network, and it is implemented using large-scale matrix and tensor operations. The calculation and application of gradients are the engine of deep learning, and they are entirely dependent on the rules of matrix calculus (Goodfellow et al., 2016).

- *Singular Value Decomposition (SVD) in Recommender Systems*

While eigen-decomposition is powerful, it is limited to square matrices. The Singular Value Decomposition (SVD) is a more general matrix factorization technique that applies to any m x n rectangular matrix. It factorizes a matrix **A** into three matrices: **A = UΣV^T**, where **U** and **V** are orthogonal matrices containing the left and right singular vectors, and **Σ** is a diagonal matrix containing the singular values (Strang, 2019). SVD is particularly powerful for identifying latent concepts within data.

This property is exploited in collaborative filtering for recommender systems. A user-item rating matrix **R** (a rectangular, and very sparse, matrix) can be factorized via a technique similar to SVD. The idea is that a low-rank approximation R ≈ P Q^T can be found, where the rows of matrix **P** represent users' affinities for a set of latent factors (e.g., genre preferences for movies), and the columns of **Q^T** represent items' expressions of those same latent factors. The dot product of a user vector and an item vector then predicts the user's rating for that item. By solving for the matrices **P** and **Q** that best approximate the known ratings, the model can fill in the missing entries of **R**, thereby predicting ratings for unrated items and generating recommendations. This demonstrates how a fundamental matrix decomposition directly enables a multi-billion dollar industry.

ISSN No:-2456-2165 https://doi.org/10.38124/ijisrt/25sep829

- *Tensors in Deep Learning*

Deep learning, particularly in domains like computer vision and natural language processing, pushes beyond the limits of matrices into the realm of tensors. An input image is not a vector; it is a 3D tensor of dimensions (height, width, channels). A batch of 32 images is a 4D tensor of dimensions (32, height, width, channels). The weights of a convolutional layer are also 4D tensors, e.g., (kernel_height, kernel_width, input_channels, output_channels) (Géron, 2022).

The operations in these networks are tensor operations. Convolution itself is a specialized tensor multiplication. The massively parallel architecture of GPUs is specifically designed to perform these tensor operations at an immense scale. Furthermore, the attention mechanism that powers the state-of-the-art Transformer architecture is built on a series of matrix multiplications (query, key, and value matrices) calculated over sequences of data, which are also represented as tensors (Vaswani et al., 2017). The entire forward and backward pass of a modern neural network is a carefully orchestrated sequence of tensor manipulations, making linear algebra (generalized to tensors) the undeniable language of deep learning.

➢ *Flowchart for Selecting the Right Linear Algebra Tool*

Given the myriad connections between linear algebra and ML, a practical guide for practitioners is valuable. The flowchart in Figure 2 provides a heuristic for selecting the appropriate linear algebraic machinery based on the primary task of the ML problem at hand. This decision-making process can streamline the model development and implementation workflow.

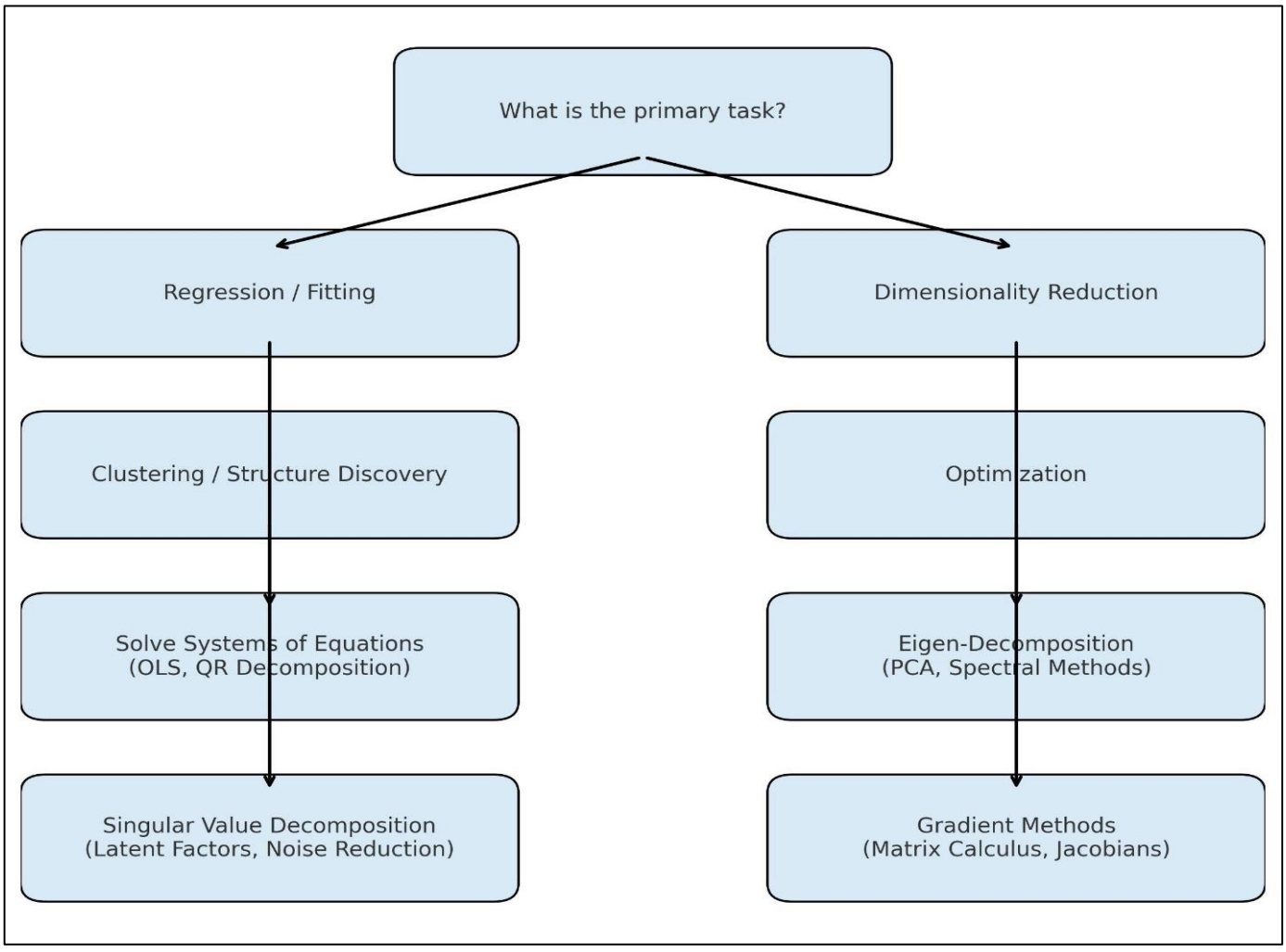

Fig 2 Flowchart for Selecting Linear Algebra Tools in Machine Learning

III. METHODS

This review was conducted through a systematic examination of existing literature at the intersection of linear algebra and machine learning. The primary sources for this synthesis included authoritative academic textbooks that establish the foundational mathematical theories, such as Strang (2019) and Goodfellow et al. (2016), alongside key research papers from major machine learning conferences like NeurIPS, ICML, and ICLR that demonstrate cutting-edge applications. Furthermore, authoritative online resources, including lecture notes from top-tier university courses (e.g., MIT, Stanford) and documentation for foundational libraries (e.g., NumPy, PyTorch), were consulted to ensure practical relevance and accuracy.

ISSN No:-2456-2165 https://doi.org/10.38124/ijisrt/25sep829

The selection criteria for sources prioritized those that provided clear, explicit links between linear algebraic concepts and specific machine learning algorithms. Clarity of explanation and foundational importance were valued over extreme novelty for the purposes of this comprehensive review. The temporal focus was on literature from 2020 to 2025 to ensure the review's contemporaneity, though seminal older works were included where necessary for foundational context. The quality of the methods described in sourced papers was assessed based on the rigor of their mathematical derivations, the clarity of their experimental design, and the validity of their statistical measures for evaluating model performance (e.g., mean squared error, accuracy, F1-score).

The information obtained from this survey was organized and categorized not by individual source, but by the primary machine learning task (e.g., data representation, regression, dimensionality reduction, optimization) and the linear algebraic concept that enables it. This organizational structure aligns with the overall goal of the paper: to provide a task-oriented guide to the linear algebra of machine learning.

IV. DISCUSSION

The synthesis presented in this review leads to an inescapable conclusion: proficiency in linear algebra is not an optional supplement to machine learning expertise but is absolutely essential for a deep, intuitive, and innovative understanding of the field. The argument is unifying: from the simplest linear regression model to the most complex transformer network, the core computations are manifestations of vector and matrix operations, matrix factorizations, and tensor calculus. Viewing ML through this lens demystifies algorithms, revealing them as sequences of understandable mathematical transformations rather than opaque black boxes.

Despite its fundamental role, a significant challenge persists: the gap between theoretical linear algebra and its practical implementation in code. The complexity of explaining advanced decompositions like SVD or the mechanics of backpropagation to newcomers can be a barrier to entry. Furthermore, the heavy abstraction provided by high-level libraries like Scikit-learn and Keras can obscure these foundations, allowing users to implement models without understanding them (VanderPlas, 2023). Bridging this gap requires pedagogical approaches that tightly couple mathematical theory with computational practice, emphasizing the "why" behind the API calls.

Future directions and opportunities in this interplay are vast. The role of numerical linear algebra libraries such as NumPy, CuPy, and the linear algebra backends of PyTorch and TensorFlow cannot be overstated; they serve as the critical bridge between theory and application, providing highly optimized implementations of these operations. The development of new hardware, such as GPUs, TPUs, and other AI accelerators, is itself an exercise in optimizing for large-scale linear algebra. These chips are specifically designed to perform matrix multiplications and convolutions with extreme efficiency and low power consumption, further cementing the symbiotic relationship between linear algebra and machine learning advancement (Jouppi et al., 2020). There is also potential for exploring more advanced or specialized matrix decompositions to inspire novel ML algorithms for graph analysis, quantum machine learning, and dynamical systems.

In conclusion, linear algebra provides the language, the tools, and the computational framework that make modern machine learning possible. It is the substrate upon which data is represented, models are constructed, learning is enacted, and innovations are built. A robust command of these principles empowers one to not just use machine learning, but to truly understand, adapt, and advance it.

REFERENCES

[1]. Bishop, C. M. (2006). *Pattern Recognition and Machine Learning*. Springer.
[2]. Deisenroth, M. P., Faisal, A. A., & Ong, C. S. (2020). *Mathematics for Machine Learning*. Cambridge University Press.
[3]. Géron, A. (2022). *Hands-On Machine Learning with Scikit-Learn, Keras, and TensorFlow* (3rd ed.). O'Reilly Media.
[4]. Goodfellow, I., Bengio, Y., & Courville, A. (2016). *Deep Learning*. MIT Press.
[5]. Jolliffe, I. T., & Cadima, J. (2016). Principal component analysis: a review and recent developments. *Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences, 374*(2065), 20150202.
[6]. Jouppi, N. P., et al. (2020). A domain-specific supercomputer for training deep neural networks. *Communications of the ACM, 63*(7), 67-78.
[7]. Mitchell, T. M. (1997). *Machine Learning*. McGraw-Hill.
[8]. Petersen, K. B., & Pedersen, M. S. (2012). The Matrix Cookbook. Technical University of Denmark.
[9]. Saxe, S. (2021). *The Mathematical Foundations of Machine Learning*. Self-published notes.
[10]. Strang, G. (2019). *Linear Algebra and Learning from Data*. Wellesley Cambridge Press.
[11]. Sutton, R. S., & Barto, A. G. (2018). *Reinforcement Learning: An Introduction* (2nd ed.). MIT Press.
[12]. Trefethen, L. N., & Bau, D. (1997). *Numerical Linear Algebra*. SIAM.
[13]. VanderPlas, J. (2023). *Python Data Science Handbook* (2nd ed.). O'Reilly Media.
[14]. Vaswani, A., et al. (2017). Attention is all you need. *Advances in Neural Information Processing Systems, 30*.