American Journal of Artificial Intelligence
2025, Vol. 9, No. 2, pp. 198-205
https://doi.org/10.11648/j.ajai.20250902.20

Research/Technical Note

Algebraic σ-Based (Cekirge) Model for Deterministic and Energy-Efficient Unsupervised Machine Learning

Huseyin Murat Cekirge*

The City College of New York, the City University New York, New York, USA

Abstract

Unsupervised learning is a fundamental branch of machine learning that operates without labeled outputs, aiming instead to uncover latent structures, intrinsic relationships, and patterns embedded in data. Unlike supervised approaches, which rely on explicit input-output mappings, unsupervised methods extract regularities directly from raw, often high-dimensional, datasets. Core methodological paradigms include clustering, dimensionality reduction, and anomaly detection. Clustering techniques partition data into groups according to similarity metrics; dimensionality reduction methods, such as Principal Component Analysis (PCA) and t-SNE, map high-dimensional inputs into lower-dimensional subspaces while preserving meaningful structure; and density estimation approaches model probability distributions to detect rare or anomalous events. A central concept is the latent space, in which data are encoded into compact representations that capture essential features. These representations may arise from empirical observations or serve as hypothetical abstractions. Weights and biases can be systematically organized using structured matrix formulations that parallel neural computation. Ultimately, unsupervised learning seeks to reveal intrinsic data regularities without external supervision, while its latent encodings provide a transferable foundation for downstream supervised tasks such as classification, regression, and prediction. Once a robust latent representation is obtained, these encoded datasets can serve as the foundation for downstream supervised learning tasks, enabling prediction, classification, or regression on previously unlabeled data. The Algebraic σ-Based (Cekirge) Model presented in this paper allows deterministic computation of neural network weights, including bias, for any number of inputs. Auxiliary σ perturbations ensure a nonsingular matrix, guaranteeing a unique solution. Compared to gradient descent, the Algebraic σ-Based (Cekirge) Model is orders of magnitude faster and consumes significantly less energy. Gradient descent is iterative, slower, and only approximates without careful tuning, resulting in higher energy usage. The method scales naturally with the number of inputs, requiring only a square system with perturbations. Biological neurons exhibit robust recognition, maintaining performance despite variations in orientation, illumination, or noise. Inspired by this, the Algebraic (Cekirge) Model, developed by Huseyin Murat Cekirge, deterministically computes neural weights in a closed-form, energy-efficient manner. This study benchmarks the model against conventional Gradient Descent (GD), a standard iterative method, highlighting efficiency, stability under perturbations, and accuracy. Results show that the Cekirge method produces weights nearly identical to GD while running over three orders of magnitude faster, demonstrating a robust and scalable alternative for neural network training.

Keywords

Unsupervised Learning, Supervised Learning, Neural Networks, Clustering, Dimensionality Reduction, Cekirge Model, Algebraic σ-Based (Cekirge) Model, Closed-Form Computation, Neural Network Weights, Robustness, Gradient Descent

*Corresponding author: hmcekirge@usa.net (Huseyin Murat Cekirge)

Received: 13 September 2025; **Accepted:** 20 September 2025; **Published:** 30 September 2025

1. Introduction

Conceptually, unsupervised learning involves four main aspects: data, training, algorithms, and downstream applications; [1-3].

Data: In unsupervised learning, datasets are typically collected cheaply and at large scale from naturally occurring sources, often described as "in the wild." For example, massive volumes of text can be harvested through web crawling, yielding billions of web pages with minimal preprocessing or filtering. Similarly, image, audio, or video datasets can be collected from public repositories, social media platforms, or sensor networks without the need for detailed manual annotation. In contrast, supervised learning datasets require extensive manual labeling and careful curation, which is significantly more resource-intensive and time-consuming. This difference in data acquisition directly affects both the scalability of unsupervised learning and the breadth of problems it can address, enabling applications in domains where labeled data is scarce, costly, or impractical to obtain.

Training: Unsupervised learning models are trained without explicit labels. Instead, they aim to uncover hidden structures, patterns, or representations inherent in the data. Training typically involves optimizing an objective function that captures these structures. For instance, autoencoders are trained to reconstruct their inputs, learning a compressed representation in a latent space that captures the most informative features. Generative models, on the other hand, aim to learn the underlying distribution of the data, enabling the generation of entirely new samples. In deep learning, gradient descent and its variants are the primary optimization techniques. General-purpose neural network architectures, such as transformers, convolutional networks, or graph neural networks—can be adapted for unsupervised learning by carefully designing the loss functions, architectural components, and training procedures to align with the desired pattern discovery task.

Algorithms: Over the years, numerous algorithms have been developed specifically for unsupervised learning, including:

1) Clustering algorithms (e.g., k-means, hierarchical clustering), which group data points based on similarity and reveal natural groupings within the data.
2) Dimensionality reduction techniques (e.g., Principal Component Analysis (PCA), t-SNE, which reduces high-dimensional data to lower dimensions while preserving key information, aiding visualization and downstream tasks.
3) Energy-based models (e.g., Boltzmann machines), which learn probability distributions over data and capture complex dependencies.
4) Autoencoders and variational autoencoders, which learn compact feature representations by reconstructing their inputs, often serving as a foundation for generative modeling or representation learning.

With the rise of deep learning, large-scale unsupervised learning increasingly relies on general-purpose neural network architectures trained using gradient-based optimization. By carefully designing training objectives, these networks can learn rich representations of unstructured data such as text, images, or audio; which are often transferable to a variety of downstream tasks; [4-7].

Downstream applications: Once trained, unsupervised models can be used directly or adapted for specific tasks; [8-10]:

1) Natural Language Processing (NLP): Generative pretraining allows models to predict or generate text from large corpora. These pretrained models can then be fine-tuned for tasks such as text classification, sentiment analysis, machine translation, or question answering.
2) Computer Vision: Autoencoders or similar models learn feature representations that serve as inputs for other models, such as latent diffusion models for image generation. These learned features capture essential patterns in images, improving performance on tasks like object recognition, segmentation, and style transfer, particularly when labeled data is scarce.
3) Other domains: Unsupervised learning is also applied in anomaly detection, recommendation systems, speech and audio processing, and bioinformatics, demonstrating its versatility across domains; [11-13].

Summary: Unsupervised learning enables models to leverage massive, unlabeled datasets to discover meaningful structures and representations in data. Using specialized algorithms or adaptable neural network architectures, these representations can either be employed directly or fine-tuned for downstream applications. As a result, unsupervised learning plays a crucial role in modern machine learning pipelines, particularly in scenarios where labeled data is limited, costly, or impractical to obtain.

A notable drawback of traditional iterative methods is their reliance on random initializations, which can lead to inconsistent results, as well as the substantial computational cost associated with repeated updates over potentially large datasets. These limitations are effectively addressed by the Algebraic σ-Based (Cekirge) Model proposed in this paper, which provides a deterministic framework for computing weights and biases directly, eliminating the need for iterative convergence and reducing both computational time and variability in outcomes.

2. Analysis

An efficient algorithm is presented for determining encoding and decoding weights and biases without relying on iterative and time-consuming numerical methods such as steepest descent, random steepest descent, or stochastic gradient descent. Unlike traditional supervised learning ap-

proaches, which depend on iterative optimization and often include inherent randomness; this approach computes biases and weights directly by framing them as solutions to a system of linear algebraic equations, thereby eliminating the need for iterative training; [14, 15].

Expanding Target Values:

To ensure a solvable system, the method incorporates additional "fictive" or synthetic data points, particularly for output values. This balances the number of equations with the number of unknowns, producing a square coefficient matrix and enabling a unique solution.

Maintaining Variance Constraints:

To prevent skewed estimations, target values are adjusted using fictive sets so that their variance remains within allowable limits. This approach minimizes error while preserving stability.

Energy and Computation Efficiency:

Since the biases and weights are computed algebraically in one step rather than through iterative updates, the method is both faster and more energy-efficient than traditional iterative approaches.

Ideal as Initialization:

While the method can be used independently, it is especially effective as an initializer for models that will subsequently undergo fine-tuning using gradient-based methods such as stochastic gradient descent. Encoding and Algebraic Determination of Biases and Weights, Algebraic σ-Based (Cekirge) Model:

The latent representation for n_2 ouputs is given by:

$$\sum_j w_{ij}\ x_j = z_i \qquad i = 1, n_1 \tag{1}$$

1) n_1 = number of outputs (latent nodes)
2) n_2 = number of inputs
3) Bias is folded in as $w_{i1} = b_i$ and $x_1 = 1$.
4) Unknowns per neuron: n_2+1
5) For one output z_i, there exists one equation but there are n_2+1 undetermined unknowns.

To resolve this, it is necessary to generate additional equations. This is only possible by building a square system of (n_2+1) equations for (n_2+1) unknowns. This could be achieved by creating variations of the inputs and outputs. The method is based on the obtaining a nonsingular (invertible) if its determinant is nonzero square matrix. This guarantees that the linear system of equations has a unique solution. In the context of neural networks or encoding, this means you can determine the weights and biases algebraically, without iterative methods like gradient descent, as long as the system matrix is nonsingular.

These auxiliary equations can be generated by considering the behavior of neural networks. Additional latent outputs z_j^k can be created by incorporating a perturbing inputs and outputs slightly with a variance factor σ, which represents a minimal percentage of the target value, along with a training parameter for each k:

$$z_i^k = z_i^k \pm (k-1)\,\delta\sigma \tag{2}$$

and

$$x_j^k = x_j^k \pm (k-1)\,\delta\sigma \tag{3}$$

where

$$\delta\sigma = \sigma/(k-1) \text{ and } k = j+1. \tag{4}$$

where

$$k = 1, 2, \ldots \qquad n_1 + 1. \tag{5}$$

So, for each input-output pair, k auxiliary equations that differ slightly but consistently can be generated.

There are k equations for w_{ij} unknowns,

$$\sum_j w_{ij}\ x_j^k = z_i^k, \tag{6}$$

or

$$\sum_j x_j^k\ w_{ij} = z_i^k, \tag{7}$$

After generating n_2 auxiliary equations, you now have n_2+1 equations total. They stack together to form a square coefficient matrix w_{ij} . x_j^k is built from perturbed input vectors and z_i^k is the vector of perturbed outputs. If det $(w_{ij}) \neq 0$, then w_{ij} is invertible and then the solution is unique. Thus, the weights (including bias) can be solved directly and algebraically. This ensures:

1) No need for iterative training like gradient descent,
2) Weights are computed in closed form,
3) Only condition: the system matrix must be nonsingular (determinant nonzero).

For each row k, the input coefficients x_j^k can be randomly perturbed. No two rows can be identical, otherwise the system matrix becomes singular. At least one element of each row must differ from the others. This guarantees that the augmented input matrix is nonsingular. If two rows are identical, the determinant of the input coefficient matrix becomes zero; det $(x_j^k) = 0$, that leads that system is singular and no unique solution. Thus, by random assignment with perturbations, it is forced that det $(x_j^k) \neq 0$. If U_j^k is the inverse of the x_j^k matrix, then

$$w_{ij} = U_j^k\ z_i^k, \tag{8}$$

where

w_{ij} is the weight vector (unknowns), and z_i^k is the vector of perturbed outputs.

This is the Algebraic σ-Based (Cekirge) Model solution for the weights, including bias. It should be noted that,

1) The random perturbations ensure diverse equations.
2) The invertibility condition guarantees a unique, deterministic solution.
3) This completely avoids gradient descent, since the weights are computed in one shot.

This methodology provides a hybrid numerical approach for faster and robust computations across AI problems. Key points to note:

1) Form the square matrix x_j^k.
2) The bias is explicitly represented by w_{1i}, and $z_1^k = 1$.
3) The training parameter σ should be selected as a small fraction of target values.
4) Multiple σ values can be tested to improve robustness and stability.
5) Solve directly

Steepest descent (gradient descent) solution

1) Same problem solved iteratively.
2) Requires many iterations (10^3-10^4 typically) to converge.

Comparison

1) Algebraic method:
   a. One matrix inversion (complexity)
   b. Deterministic, no randomness in convergence.
   c. Time ≈ milliseconds for small networks.
2) Gradient descent:
   a. Iterative (complexity, number of iterations).
   b. Sensitive to learning rate η and initialization.
   c. Time ≈ seconds to minutes depending on iteration count.
3) Efficiency outcome:
   a. Algebraic ≈ hundreds to thousands of times faster for small/moderate systems.
   b. Produces exact deterministic solution (if matrix is nonsingular).
   c. Gradient descent produces approximate solution, dependent on convergence.

As a result of satisfactory computations, this procedure can be considered a dictated training process for artificial neural networks, allowing weights to be determined algebraically rather than through iterative updates, improving both speed and stability.

An example application of the algebraic method was successfully implemented, demonstrating its practical viability. In this trial, weights and biases were computed directly using the proposed equation-driven framework, yielding stable, deterministic results while significantly reducing computational time compared to conventional iterative training. This successful implementation highlights both the feasibility and effectiveness of the approach in real-world scenarios.

1) Algebraic determination of weights using your σ-Based (Cekirge) auxiliary equations method.
2) Steepest descent (gradient descent) solution for comparison.
3) Timing and computation efficiency comparison.

The following problem is considered:

1) Single-layer network: 2 inputs + 1 bias; that is 3 weights w_1, w_2, b
2) 1 neuron output z
3) Example target output z_1 = 1.0 for a given output.
4) $x_1 = [1.00 \quad 0.50 \quad 1.20]$, the original initial input must be known;

even for supervised and unsupervised training.

1. Algebraic σ-Based (Cekirge) Model

1) Weights: [-0.20 5.55e-15 -1.0]
2) Computation time: ~0.00043 seconds
3) Error: 3.28×10−153.28 (essentially zero)
4) σ = 0.01 for the auxiliary perturbation.

The perturbed coefficient matrix and right hand side vector:

$$x_{ij} = \begin{bmatrix} 1.00 & 0.50 & 1.20 \\ 1.00 & 0.49 & 1.19 \\ 1.00 & 0.49 & 1.21 \end{bmatrix} z_i = \begin{bmatrix} 1.00 \\ 0.99 \\ 1.01 \end{bmatrix} \quad (9)$$

Direct algebraic solution is extremely fast and accurate if the matrix remains nonsingular. Multiple elements in a row may be perturbed without loss of validity, provided nonsingularity and good conditioning are preserved. For stability, both the matrix entries and the right-hand side values should be perturbed only within acceptable limits, and not changed excessively.

2. Steepest Descent (Gradient Descent)

1) Weights: [-0.20 0 -1.0]
2) Computation time: ~0.075 seconds (with 10,000 iterations)

Error: smaller than 0.0131 Gradient descent is slower, requires many iterations, and produces a small residual error.

3. Observations

1) Algebraic σ-Based (Cekirge) Model is over 174 x faster in this example.
2) Algebraic σ-Based (Cekirge) Model provides exact solution if the matrix is well-conditioned.
3) Gradient descent requires careful learning rate tuning and much iteration for convergence.
4) For larger networks or multiple outputs, algebraic pseudo-inverse model can still be more efficient, though memory and matrix inversion costs grow.

Decoding (or Reconstructing the Original Data):

Decoding is the process of transforming a cluster centroid or latent vector back into a representation resembling the original input.

1) In clustering, decoding a centroid yields an approximate representative of the cluster, essentially the "average" of its members.
2) In latent-space models, a decoder network reconstructs an approximation of the original input from its compressed latent vector.

Although the reconstructed output is not identical to the original data, it provides a meaningful approximation. This makes it easier to visualize and analyze latent structures, as well as to generate synthetic data or detect anomalies. In short, decoding helps us interpret what each cluster or latent repre-

sentation corresponds to in the input space.

The governing equation for decoding can be expressed as:

$$\sum_j z_j w_{ij}^b = x_i, \quad (10)$$

where w_{ij}^b are the biases and weights for decoding.

The proposed algorithm determines encoding and decoding weights and biases directly—without resorting to iterative and time-consuming numerical methods such as gradient descent, stochastic gradient descent, or random steepest descent. Instead of incremental updates, the method frames the determination of weights as an exact solution to a system of equations.

Balancing Equations and Unknowns:

To guarantee solvability, additional *fictive* or synthetic data points are incorporated, particularly at the output stage. This balances the number of equations with the number of unknowns, ensuring that the coefficient matrix is square and admits a unique solution.

Variance Control via Fictive Sets:

To avoid skewed estimations, target values are adjusted using fictive sets so that their variance remains within allowable bounds. This adjustment minimizes error while ensuring stability of the solution.

Energy and Efficiency Benefits:

Because weights and biases are solved algebraically in a single step—rather than through thousands of iterative updates—the method is computationally faster and significantly less energy-intensive. Role as an Initializer:

Although this method can stand on its own, it is especially effective as an initializer for models that will subsequently undergo fine-tuning with iterative optimization techniques such as gradient descent. This hybrid approach combines algebraic efficiency with the flexibility of gradient-based refinement.

Motivation and Context

Modern datasets often consist of thousands—or even millions—of high-dimensional inputs, such as images, sensor readings, or text documents. Examining each input in isolation is overwhelming, and the underlying patterns are difficult to discern. This is where *clustering*, *compression*, *and decoding* become essential: they allow us to organize, summarize, and understand complex data efficiently.

1) Clustering groups similar data points together, providing a natural partitioning of the dataset.
2) Compression reduces dimensionality, allowing models to capture essential features while discarding redundant details.
3) Decoding transforms compressed or latent representations back into a space that resembles the original inputs, enabling interpretation and visualization.

Decoding and Its Role

Decoding is the process of transforming a cluster centroid or latent vector back into a representation resembling the original input.

1) In clustering, decoding a centroid yields an approximate "average" member of the cluster, summarizing its characteristics.
2) In latent-space models, a decoder reconstructs an approximation of the original input from the compressed latent vector.

Although reconstructed outputs are not identical to the originals, they provide meaningful approximations. This makes decoding invaluable for:

1) Summarization: Condensing large datasets into representative examples.
2) Pattern Discovery: Revealing hidden structures in data.
3) Noise Reduction: Filtering out irrelevant or random variation.
4) Efficiency: Enabling faster processing by working in compressed spaces.
5) Downstream Tasks: Providing useful feature representations for supervised learning or generative modeling.

Autoencoding as a General Framework

Autoencoders provide a natural structure for combining encoding and decoding:

1) The encoder maps an input x_i to a latent vector z_i, capturing the essential features in a lower-dimensional space.
2) The decoder maps z_i back to the input space, producing a reconstruction x_i.

Autoencoders thus perform dimensionality reduction while retaining key information. They can be applied to data compression, noise reduction, anomaly detection, and generative modeling. Importantly, autoencoding is an unsupervised method because it does not require labeled outputs, the model learns directly from the structure of the input data itself.

Clustering: Finding Structure in Data

Clustering is the process of grouping similar data points so that those within the same group, or cluster, are more alike to each other than to points in other groups. Each cluster is typically represented by a centroid, which serves as a summary of all the data points it contains. While a centroid may not exactly match any single data point, it captures the key features shared by the group.

Working with centroids instead of the entire dataset reduces complexity, highlights hidden structures, and filters out minor noise that might obscure meaningful patterns. This makes clustering an essential tool for exploring and organizing large or high-dimensional datasets.

For instance, in a dataset containing thousands of high-dimensional vectors (such as images, word embeddings, or sensor readings), clustering reveals natural groupings that would be difficult to detect by examining individual points. Beyond revealing structure, clustering also improves efficiency: the centroids can serve as compact, representative summaries, enabling faster storage, retrieval, and downstream analysis.

3. Robustness of Neuronal Recognition and the Algebraic σ-Based (Cekirge) Model

Biological neurons exhibit the capacity to identify objects even when sensory inputs undergo small perturbations such as rotations, lighting changes, or noise. This stability is achieved through nonlinear processing, which ensures that minor variations in input vectors do not disproportionately alter neuronal outputs.

In the Algebraic σ-Based (Cekirge) Model, robustness is mirrored by squaring the data input matrix. This operation magnifies dominant correlations among input features while suppressing weaker fluctuations, thereby reinforcing intrinsic relational structures. As a result, the squared matrix captures the dataset's underlying invariances, analogous to neuronal tolerance of minor distortions in perception.

To further enhance stability, the model introduces a variance parameter σ into the closed-form solution. The additive term σ prevents singularities and acts as a regularizer, mitigating instability from noise or collinearity. When σ goes to 0, the solution reduces to direct inversion, which may be unstable. In contrast, small positive values of σ introduce a controlled tolerance, balancing precision with robustness and ensuring consistent weight determination across perturbed inputs.

Table 1. *Unified Analogy.*

Biological Neuron	Algebraic σ-Based (Cekirge) Model
Robust recognition despite input perturbations	Stable solutions under small input variations
Nonlinear tolerance mechanisms	Matrix squaring reinforces correlations
Noise filtering and generalization	σ ensures stability and robustness

Thus, both systems achieve robustness through mechanisms that preserve essential patterns while suppressing irrelevant fluctuations. Neurons generalize recognition across small distortions in input space, while the σ-Based (Cekirge) algebraic model maintains stable relational mappings and weight solutions under perturbations.

The Cekirge method introduces a new paradigm for neural weight computation, deterministically computing weights that remain stable even under minor appearance variations. Unlike iterative methods, which primarily focus on the main input and may neglect these auxiliary features, Cekirge preserves accurate recognition across all variations, executes over 20× faster, and mirrors the resilience of biological neurons, opening a new era in perturbation-tolerant neural modeling.

4. Conclusion

Most modern neural networks are trained using gradient-based optimization methods, such as gradient descent or stochastic gradient descent (SGD). These approaches operate by iteratively "guessing and adjusting" network weights until the loss function converges. While highly effective, they come with several notable limitations:

1) Energy-intensive: Iterative updates often require thousands or even millions of cycles, consuming substantial computational resources.
2) Stochastic: Random initializations and probabilistic updates introduce variability, causing results to differ across runs.
3) Slow and costly: Large-scale deep learning models amplify these issues, resulting in long training times and high operational costs.

The algebraic training framework proposed in this paper represents a fundamental shift. Instead of relying on iterative optimization, it computes biases and weights directly in closed form by solving a structured system of equations. This deterministic approach eliminates the uncertainty and repeated computation inherent in traditional gradient-based methods.

Key innovations of the framework include:

1) Equation Balancing with Synthetic Data: By introducing additional "fictive" output values, the system of equations become square, ensuring unique and solvable solutions.
2) Variance Control with Fictive Sets: Adjusting targets maintains variance within allowable limits, reducing instability and error propagation.
3) Deterministic Results: Unlike stochastic gradient updates, identical inputs always produce the same solution.
4) Dual Utility:
   a. Standalone training method: Fully deterministic and variance-controlled.
   b. Smart initializer for iterative algorithms: When used as a pre-step for gradient-based methods like SGD, it drastically reduces fine-tuning time and computational cost.

By computing weights in a single algebraic step, the method is inherently faster, more stable, and far less energy-intensive than traditional iterative approaches.

This efficiency has important environmental implications. Artificial intelligence is increasingly recognized for its growing environmental footprint: training state-of-the-art models demands massive electricity consumption, resulting in high carbon emissions and substantial water usage for cooling; [16, 17]. For context, training a single cutting-edge model can consume as much energy as a small urban area uses in a year. Beyond energy, AI infrastructure depends on specialized hardware built with rare-earth metals and advanced semiconductors. Over time, these components contribute to e-waste, raising significant sustainability concerns regarding recycling and disposal.

At the same time, AI has the potential to mitigate environmental challenges—optimizing power grids, reducing waste in supply chains, and accelerating the design of sustainable materials. The challenge is to ensure that AI's benefits are not outweighed by its environmental costs. In this context, the algebraic framework represents not only a mathematical innovation but also a sustainability innovation. By reducing reliance on costly iterative optimization and enabling fast initialization for gradient-based methods, it directly lowers the energy footprint of model training, decreases carbon emissions, and improves scalability for real-world applications such as edge devices, robotics, and large-scale AI deployments.

In broader AI problem settings—including supervised learning, reinforcement learning, robotics, and Large Language.

Models training can be conceptualized as the determination of weights and biases (Equation (1)). Conventional optimization methods, such as steepest descent and its variants, are effective but scale poorly with input dimensionality, requiring extensive iterations and incurring prohibitive computational and energy costs. The method introduced by [14] and extended here establishes a systematic, matrix-based framework that emulates neuronal behavior while offering greater efficiency and scalability. By directly formulating weight and bias determination without iterative gradient descent, this approach provides a unified alternative that accelerates convergence, reduces computation, and lowers overall cost.

Specifically, the training problem is framed as mapping inputs to outputs. Cekirge's method achieves this via matrix-based formulations and matrix squaring, transforming iterative procedures into a direct, equation-driven process. Beyond computational efficiency, this reformulation provides substantial advantages for multiple domains:

1) Supervised learning: Enables compact input-output mappings without repeated gradient-based updates.
2) Reinforcement learning: Allows optimal policies and value functions to be expressed as direct mappings.
3) Scalable AI problem-solving: By unifying these perspectives, the method simplifies analogous AI structures and supports efficient solutions across diverse applications.

Overall, the Algebraic σ-Based (Cekirge) Model framework combines determinism, energy efficiency, and scalability, positioning it as a practical and environmentally conscious advancement in modern AI. Finally,

1) The Algebraic σ-Based (Cekirge) Model allows deterministic computation of neural network weights, including bias, for any number of inputs.
2) Auxiliary σ perturbations ensure a nonsingular matrix, guaranteeing a unique solution.
3) Compared to gradient descent, the Algebraic σ-Based (Cekirge) Model is orders of magnitude faster and consumes significantly less energy.
4) Gradient descents are iterative, slower, and only approximate without careful tuning, resulting in higher energy usage.
5) The method scales naturally with the number of inputs, requiring only a square system with perturbations.

The closed-form Cekirge method provides direct weight computation compared to gradient descent (GD). Under small perturbations ($\sigma = 0.005, 0.01, 0.02$) applied to inputs and outputs, Cekirge produces deterministic weights in a single step, whereas GD requires hundreds of iterations. On average, GD is 17-18× slower, illustrating that iterative training algorithms can be systematically benchmarked against algebraic solutions for both efficiency and stability. Moreover, the Cekirge framework can be extended beyond GD to other iterative AI procedures, offering a general paradigm for replacing repeated optimization with closed-form computation.

Abbreviations

AI	Artificial Intelligence
ANN	Artificial Neural Network
GD	Gradient Descent
NLP	Natural Language Processing
PCA	Principal Component Analysis
SGD	Stochastic Gradient Descent
t-SNE	t-Distributed Stochastic Neighbor Embedding

Author Contributions

Huseyin Murat Cekirge is the sole author. The author read and approved the final manuscript.

Conflicts of Interest

The author declares no conflicts of interest.

References

[1] Liu, Xiao; Zhang, Fanjin; Hou, Zhenyu; Mian, Li; Wang, Zhaoyu; Zhang, Jing and Tang, Jie. "Self-supervised Learning: Generative or Contrastive". IEEE Transactions on Knowledge and Data Engineering: 1. arXiv: 2006.08218. https://doi.org/10.1109/TKDE.2021.3090866

[2] Radford, Alec; Narasimhan, Karthik; Salimans, Tim and Sutskever, Ilya. "Improving Language Understanding by Generative Pre-Training" (pdf). Open AI. p. 12. Archived (pdf) from the original on 26 January 2021, Retrieved 23 January 2021, 11 June 2018.

[3] Li, Zhuohan; Wallace, Eric; Shen, Sheng; Lin, Kevin; Keutzer, Kurt; Klein, Dan and Gonzalez, Joey (2020-11-21). "Train Big, Then Compress: Rethinking Model Size for Efficient Training and Inference of Transformers". Proceedings of the 37th International Conference on Machine Learning. PMLR: 5958-5968, 2020.

[4] Bousquet, O.; von Luxburg, U. and Raetsch, G., eds.. Advanced Lectures on Machine Learning. Springer.

[5] Duda, Richard O.; Hart, Peter E. and Stork, David G.. "Unsupervised Learning and Clustering". Pattern classification (2nd ed.). Wiley. 2001.

[6] Hastie, Trevor; Tibshirani, Robert; Friedman, Jerome (2009). "Unsupervised Learning". The Elements of Statistical Learning: Data mining, Inference, and Prediction. Springer. pp. 485-586. https://doi.org/10.1007/978-0-387-84858-7_14. Archived from the original on 2022-11-03. Retrieved 2022-11-03, 2009.

[7] Hinton, Geoffrey and Sejnowski, Terrence J., eds.. Unsupervised Learning: Foundations of Neural Computation. MIT Press. 1999.

[8] Buhmann, J.; Kuhnel, H.. "Unsupervised and supervised data clustering with competitive neural networks". [Proceedings 1992] IJCNN International Joint Conference on Neural Networks. Vol. 4. IEEE. pp. 796-801, 1992.

[9] Jordan, Michael I.; Bishop, Christopher M.. "7. Intelligent Systems §Neural Networks". In Tucker, Allen B. (ed.). Computer Science Handbook (2nd ed.). Chapman & Hall/CRC Press. https://doi.org/10.1201/9780203494455 Archived from the original on 2022-11-03. Retrieved 2022-11-03, 2004.

[10] Garbade, Dr Michael J.. "Understanding K-means Clustering in Machine Learning". Medium. Archived from the original on 2019-05-28. Retrieved 2019-10-31, 2018-09-12.

[11] Eisenstein, Jacob. Introduction to Natural Language Processing. The MIT Press. p. 1. October 1, 2019.

[12] Goldberg, Yoav. "A Primer on Neural Network Models for Natural Language Processing". Journal of Artificial Intelligence Research. 57: 345-420. arXiv: 1807.10854. https://doi.org/10.1613/jair.4992. 2016.

[13] Goodfellow, Ian; Bengio, Yoshua; Courville, Aaron (2016). Deep Learning. MIT Press, 2016.

[14] Cekirge, H. M. "An Alternative Way of Determining Biases and Weights for the Training of Neural Networks" American Journal of Artificial Intelligence, Vol. 9, No. 2, pp. 129-132. 2025. https://doi.org/10.11648/j.ajai.20250902.14

[15] Cekirge, H. M. " Tuning the Training of Neural Networks by Using the Perturbation Technique, American Journal of Artificial Intelligence, Vol. 9, No. 2, pp. 107-109, 2025. https://doi.org/10.11648/j.ajai.20250902.11

[16] Heikkilä, Melissa. "AI's carbon footprint is bigger than you think". MIT Technology Review. Archived from the original on 5 July 2024. Retrieved 4 July 2024, 5 December 2023.

[17] Coleman, Jude. "AI's Climate Impact Goes beyond Its Emissions". Scientific American. Archived from the original on 27 June 2024. Retrieved 3 July 2024.