

Kolmogorov-Arnold Networks (KAN) are inspired by the Kolmogorov-Arnold representation theorem, which provides a way to express any multivariate continuous function as a composition of univariate functions. This leads to a neural network design that aims to achieve efficient and theoretically robust approximations of complex functions. This blog will provide a mathematical overview of Kolmogorov-Arnold Networks, how they differ from standard Neural Networks, and how they relate to Kolmogorov-Arnold Machines (KAM). The focus will be on the core mathematical concepts underlying each model.

1. Standard Neural Networks (NN)

Mathematical Basis

Standard Neural Networks are designed to approximate complex functions ($f: \mathbb{R}^d \rightarrow \mathbb{R}^m$) by learning from data:

Input Layer:

$[X = \{x_1, x_2, \dots, x_n\} \in \mathbb{R}^d]$ where (X) represents the input data.

Hidden Layers:

Each hidden layer computes: $[h^{(l)} = \sigma(W^{(l)} h^{(l-1)} + b^{(l)})]$ where:

- $(h^{(l)})$ is the activation of layer (l) ,
- $(W^{(l)} \in \mathbb{R}^{d_{l-1} \times d_l})$ is the weight matrix,
- $(b^{(l)} \in \mathbb{R}^{d_l})$ is the bias,
- (σ) is a non-linear activation function (like ReLU or sigmoid).

Output Layer:

$[\hat{y} = \sigma(W^{(L)} h^{(L-1)} + b^{(L)})]$ where (\hat{y}) is the output.

Universal Approximation:

Standard NNs rely on the Universal Approximation Theorem, which states that a single hidden layer NN with sufficient neurons can approximate any continuous function (f) over a compact set.

Key Characteristics:

- Data-driven.
- Relies on depth and non-linearities for function approximation.
- Weights and biases are learned during training to minimize a loss function.

2. Kolmogorov-Arnold Machines (KAM)

Kolmogorov-Arnold Theorem

The Kolmogorov-Arnold theorem states that any continuous multivariate function ($f: \mathbb{R}^d \rightarrow \mathbb{R}$) can be decomposed as: $[f(x_1, x_2, \dots, x_d) = \sum_{q=0}^{2d} \phi_q(\sum_{p=1}^d \psi_{pq}(x_p))]$ where:

- (ψ_q) and (ψ_{pq}) are continuous univariate functions.
- (d) is the dimension of the input space.

This representation implies that any complex function can be approximated by a sum of univariate functions, allowing KAM to utilize simpler computations compared to standard NNs.

Mathematical Integration in KAM

In KAM, the network architecture is designed to match the Kolmogorov-Arnold representation by separating the multivariate function approximation into:

1. **Inner Layer:** $[s_q = \sum_{p=1}^d \psi_{pq}(x_p)]$ where (s_q) is a linear combination of inputs after being transformed by univariate functions (ψ_{pq}) .
2. **Outer Layer:** $[f(x) = \sum_{q=0}^{2d} \phi_q(s_q)]$ where each (s_q) is passed through another univariate function (ϕ_q) , which are summed to give the final output.

Key Characteristics:

- Relies on univariate transformations.
- Simplifies complex multivariate problems by decomposing them.
- Highly interpretable due to its explicit function decomposition.

3. Kolmogorov-Arnold Networks (KAN)

Core Mathematical Idea

KAN builds upon the Kolmogorov-Arnold framework but introduces a neural network implementation that leverages deep architectures. Instead of using predefined univariate functions (ψ_{pq}) and (ϕ_q) , KAN trains these functions using neural networks.

Mathematical Formulation in KAN

KAN is structured as follows:

Inner Univariate Transformations:

$[s_q = \sum_{p=1}^d \psi_{pq}(x_p)]$ where:

- Each (ψ_{pq}) is modeled as a small neural network, taking (x_p) as input: $[\psi_{pq}(x_p) = \sigma(W_{pq} x_p + b_{pq})]$
- The parameters (W_{pq}) and (b_{pq}) are learned during training.

Outer Univariate Aggregation:

$[\hat{f}(x) = \sum_{q=0}^{2d} \phi_q(s_q)]$ where:

- Each (ϕ_q) is also a small neural network: $[\phi_q(s_q) = \sigma(W_q s_q + b_q)]$

Loss Function:

The loss function for training is similar to traditional neural networks:
$$\mathcal{L}(\hat{y}, y) = \sum_i \mathcal{L}_i(\hat{y}_i, y_i)$$
 where the predicted output ($\hat{y}_i = \hat{f}(x_i)$) is based on the KAN architecture.

Optimization:

Gradient-based methods are used to update the weights (W_{pq}), (b_{pq}), (W_q), and (b_q) simultaneously:
$$W \leftarrow W - \eta \frac{\partial \mathcal{L}}{\partial W}$$
 where (W) collectively represents all weights in the network.

Key Characteristics:

- Utilizes neural networks to approximate univariate functions.
- Keeps the decomposition structure of KAM while using deep learning for flexibility.
- Can handle higher-dimensional data with deep architectures.

Comparison of KAN, Standard NN, and KAM

Aspect	Standard NN	KAM	KAN
Function Approximation	Universal Approximation Theorem	Kolmogorov-Arnold Decomposition	Kolmogorov-Arnold Decomposition with NN
Complexity Handling	Depth + Non-linearity	Univariate Function Decomposition	NN-based Univariate Function Decomposition
Interpretability	Low (black-box)	High (explicit decomposition)	Moderate (NN-driven decomposition)
Mathematical Structure	Matrix operations with non-linearity	Univariate function sums	Hybrid (NN for univariate functions)
Training	Data-driven	Fixed structure with univariate adjustments	Data + NN-driven univariate learning
Parameters	Weight matrices	Parameters for univariate functions	Neural networks for (ϕ_q) and (ψ_{pq})