

Introducción

Inspirados en el teorema de representación de Kolmogorov–Arnold, Ziming Liu propone las Kolmogorov–Arnold Networks (KANs) como una alternativa a los Multi-Layer Perceptron (MLP). A diferencia de las MLP, que emplean funciones de activación fijas en cada nodo (“neurona”), las KAN incorporan funciones de activación aprendibles en cada arista (“peso”) parametrizadas por funciones univariadas splines.

Theorem 10.3.1 (Kolmogorov) *Any continuous function $f(x_1, \dots, x_n)$ defined on I_n , $n \geq 2$, can be written in the form*

$$f(x_1, \dots, x_n) = \sum_{j=1}^{2n+1} \chi_j \left(\sum_{i=1}^n \psi_{ij}(x_i) \right),$$

where χ_j, ψ_{ij} are continuous functions of one variable and ψ_{ij} are monotone functions which are not dependent on f .

Figura 1. Teorema de representación de Kolmogorov–Arnold de Deep Learning Architectures, Calin [1]

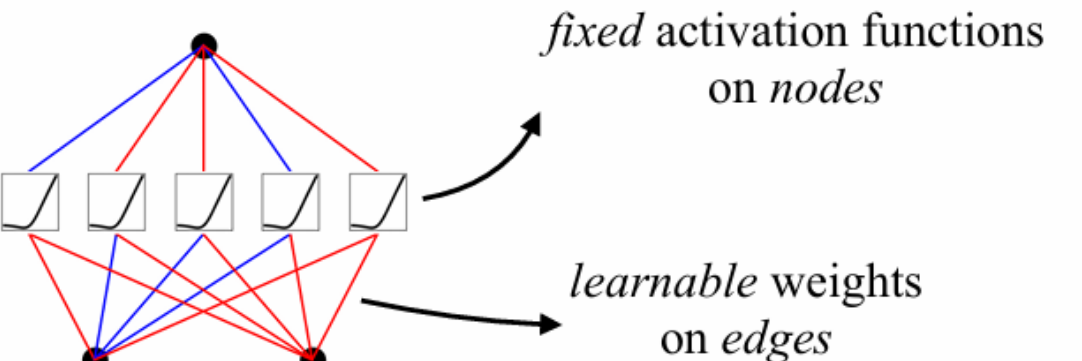
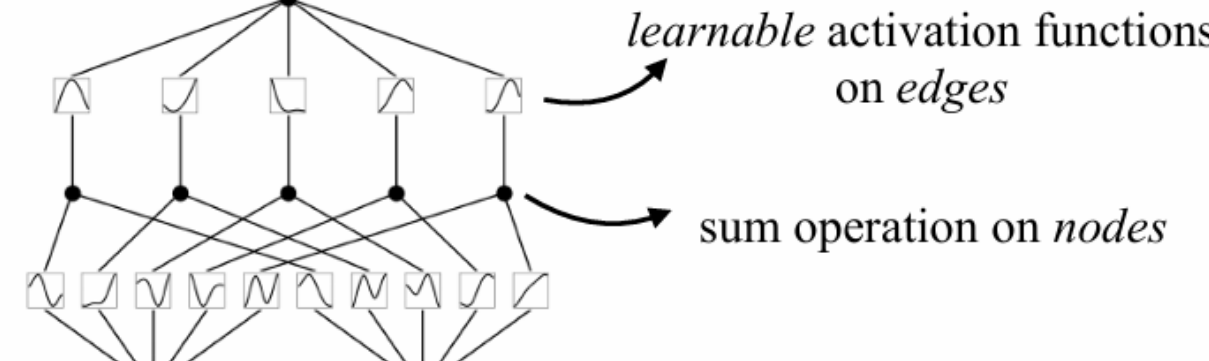
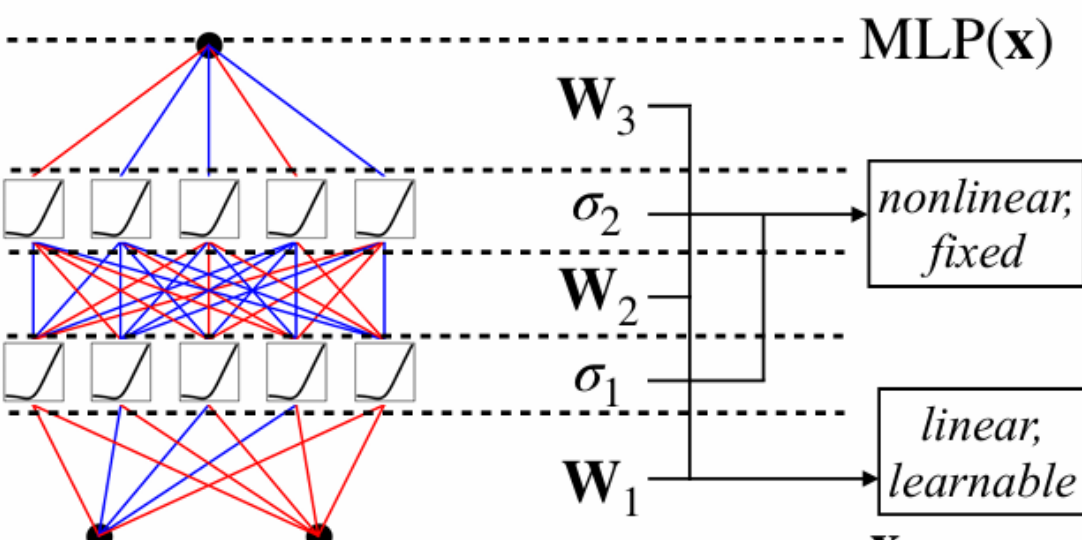
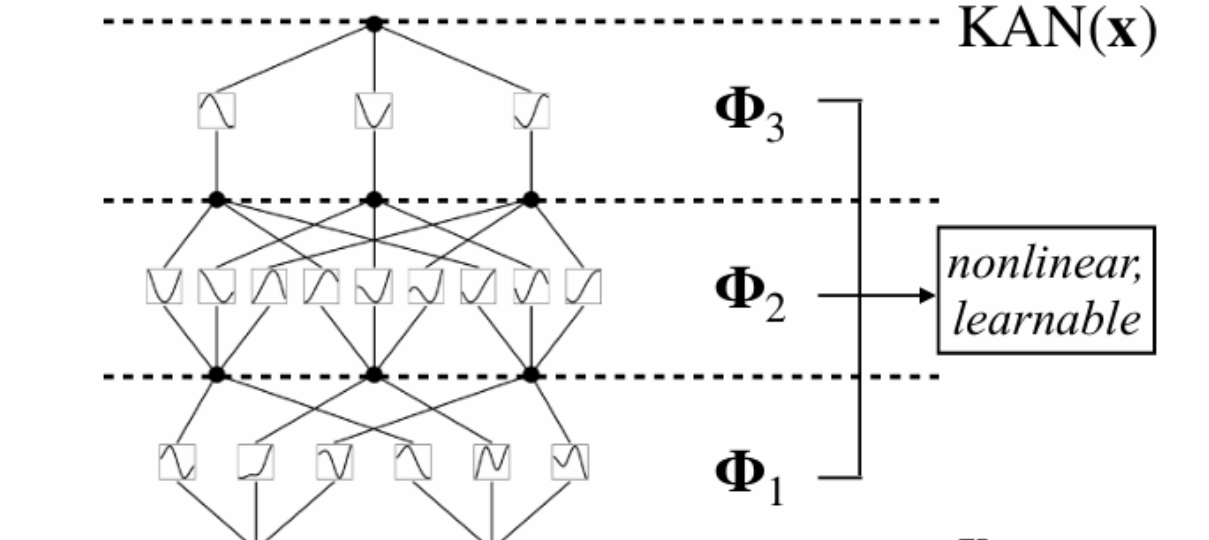
Model	Multi-Layer Perceptron (MLP)	Kolmogorov-Arnold Network (KAN)
Theorem	Universal Approximation Theorem	Kolmogorov-Arnold Representation Theorem
Formula (Shallow)	$f(\mathbf{x}) \approx \sum_{i=1}^{N(c)} a_i \sigma(\mathbf{w}_i \cdot \mathbf{x} + b_i)$	$f(\mathbf{x}) = \sum_{q=1}^{2n+1} \Phi_q \left(\sum_{p=1}^n \phi_{q,p}(x_p) \right)$
Model (Shallow)	(a) 	(b) 
Formula (Deep)	$\text{MLP}(\mathbf{x}) = (\mathbf{W}_3 \circ \sigma_2 \circ \mathbf{W}_2 \circ \sigma_1 \circ \mathbf{W}_1)(\mathbf{x})$	$\text{KAN}(\mathbf{x}) = (\Phi_3 \circ \Phi_2 \circ \Phi_1)(\mathbf{x})$
Model (Deep)	(c) 	(d) 

Figura 2. MLPs vs KANs

Detalles de implementación

Hiperparámetros:

- k orden de los B-splines;
- G grilla;
- Learning rate;
- Arquitectura de la red $L, n_0, \dots, n_{L-1}, n_L$, entre otros.

Función de activación residual: $\phi(x) = w_b b(x) + w_s \text{spline}(x)$ donde $b(x) = x/(1 + e^{-x})$

Inicialización para cada ϕ :

- w_b, w_s igual que MLP
- $\text{spline}(x) \approx 0$

Actualización de la grilla: Figura 3

Regularización: L1 norm

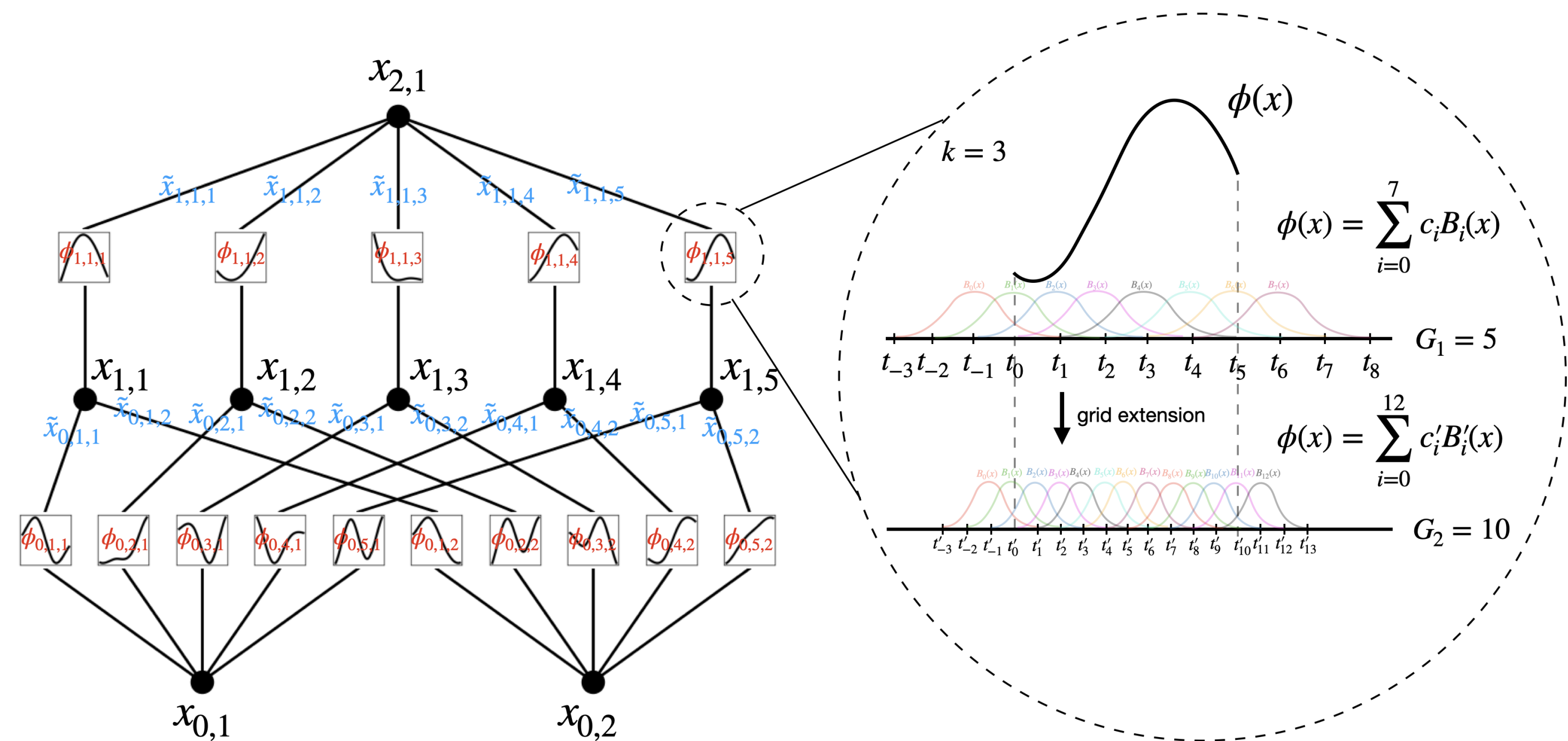


Figura 3. B-splines

Parámetros entrenables:

- $\{c_i\}_i, w_b$ y w_s , para cada función de activación

Como cada operación es diferenciable podemos aplicar descenso del gradiente (Diferenciación Automática Reversa con su respectivo grafo computacional).

Implementaciones:

- Original del autor <https://github.com/KindXiaoming/pykan>
- Eficiente en PyTorch <https://github.com/Blealtan/efficient-kan>

Teorema de aproximación

Sea $f : [0, 1]^{n_0} \rightarrow \mathbb{R}$ tal que admite una representación $f := (\Phi_{L-1} \circ \Phi_{L-2} \circ \dots \circ \Phi_1 \circ \Phi_0) \mathbf{x}$, donde $\phi_{l,i,j} \in \mathbb{C}^{k+1}([0, 1])$, entonces:

Dado G grilla, $\exists C > 0$ y $\phi_{l,i,j}^G$ B-spline de orden k (\equiv degree) tal que para cada $0 \leq m \leq k$:

$$\left\| f - (\Phi_{L-1}^G \circ \Phi_{L-2}^G \circ \dots \circ \Phi_1^G \circ \Phi_0^G) \mathbf{x} \right\|_{C^m} \leq C G^{m-(k+1)}.$$

Donde $\|g\|_{C^m} := \max_{0 \leq \beta \leq m} \sup_{x \in [0, 1]^n} |D^\beta g(x)|$. mide la magnitud de las derivadas hasta el orden m .

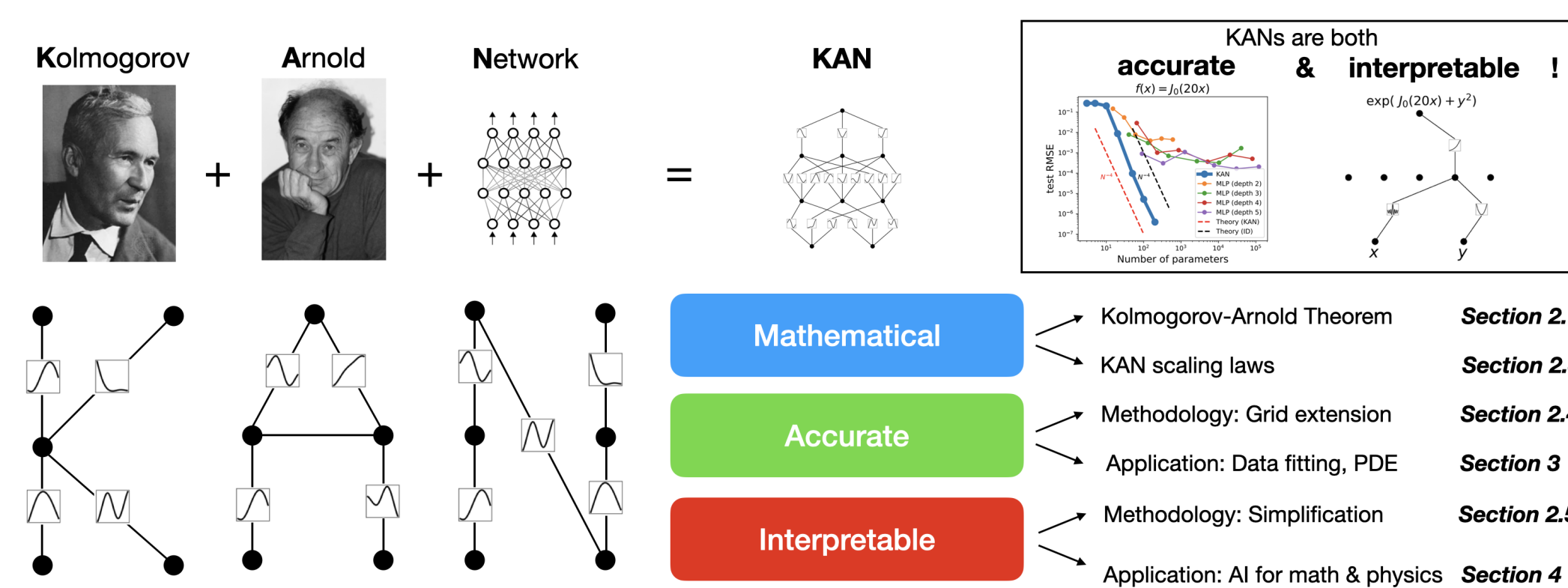
Observaciones:

- $m - (k + 1) < 0$, si G grande $G^{m-(k+1)} \sim \epsilon$
- si $m = 0, \|\cdot\|_{C^m} = \|\cdot\|_\infty$
- $\Phi_l^G : \mathbb{R}^{n_l} \rightarrow \mathbb{R}^{n_{l+1}}$ es como una matriz de activación $l = 0, \dots, L-1$ compuesta por ϕ^G 's

$$f := \left(\sum_{i_{L-1}=1}^{n_{L-1}} \phi_{L-1, i_{L-1}} \left(\sum_{i_{L-2}=1}^{n_{L-2}} \left(\dots \left(\sum_{i_2=1}^{n_2} \phi_{2, i_2} \left(\sum_{i_1=1}^{n_1} \phi_{1, i_1} \left(\sum_{i_0=1}^{n_0} \phi_{0, i_0}(x_{i_0}) \right) \right) \right) \right) \right) \right)_{i_L=1, \dots, n_L}$$

Algunos resultados o experimentos

Las redes Kolmogorov-Arnold propuestas deben su nombre a los matemáticos Andrey Kolmogorov and Vladimir Arnold. Al inspirarse en el teorema de Kolmogorov–Arnold, pueden aproximar con alta precisión la función objetivo mitigando efectos adversos como el de la maldición de la dimensionalidad. Podemos hablar de técnicas como Grid Extension, y regularización, para aumentar su precisión.



Cuando se compara el rendimiento de las redes KAN con las MLP, puede notarse un error (RMSE) menor en el primer tipo, con éste decreciendo más rápidamente mientras se aumenta el número de parámetros. Esto se ve al queremos entrenar algunas funciones a modo de ejemplo:

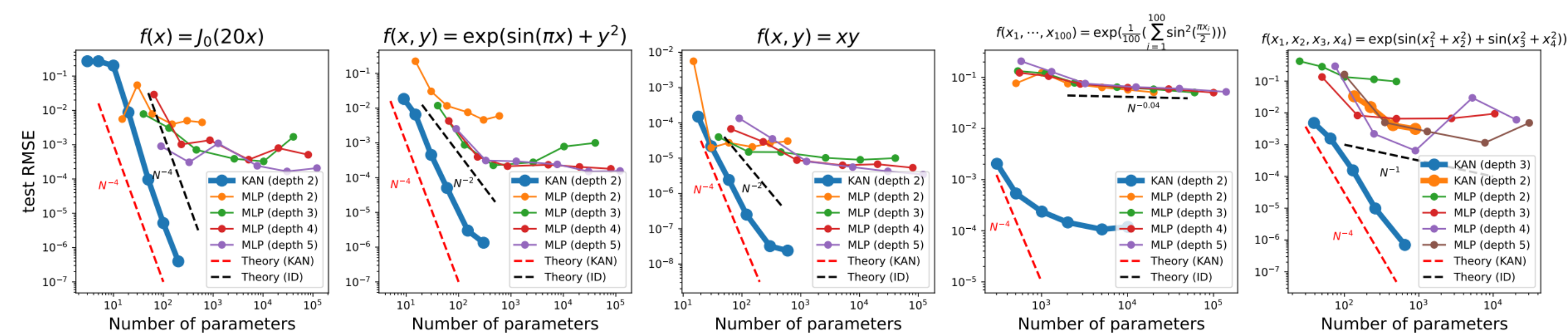


Figura 4. Comparación entre KANs y MLPs con 5 ejemplos simples

El catastrophic forgetting es un problema en el aprendizaje automático actual. Cuando una red neuronal se entrena en la tarea 1 y luego se traslada para entrenarse en la tarea 2, pronto olvida cómo realizar la tarea 1. La mayoría de redes neuronales artificiales, incluidas las MLPs, lo sufren. Las KANs pueden eludir este fenómeno aprovechando la localidad de los splines. Las MLPs suelen usar activaciones globales (ReLU, Tanh, SiLU, etc.), de modo que cualquier cambio local puede propagarse y afectar la información almacenada.

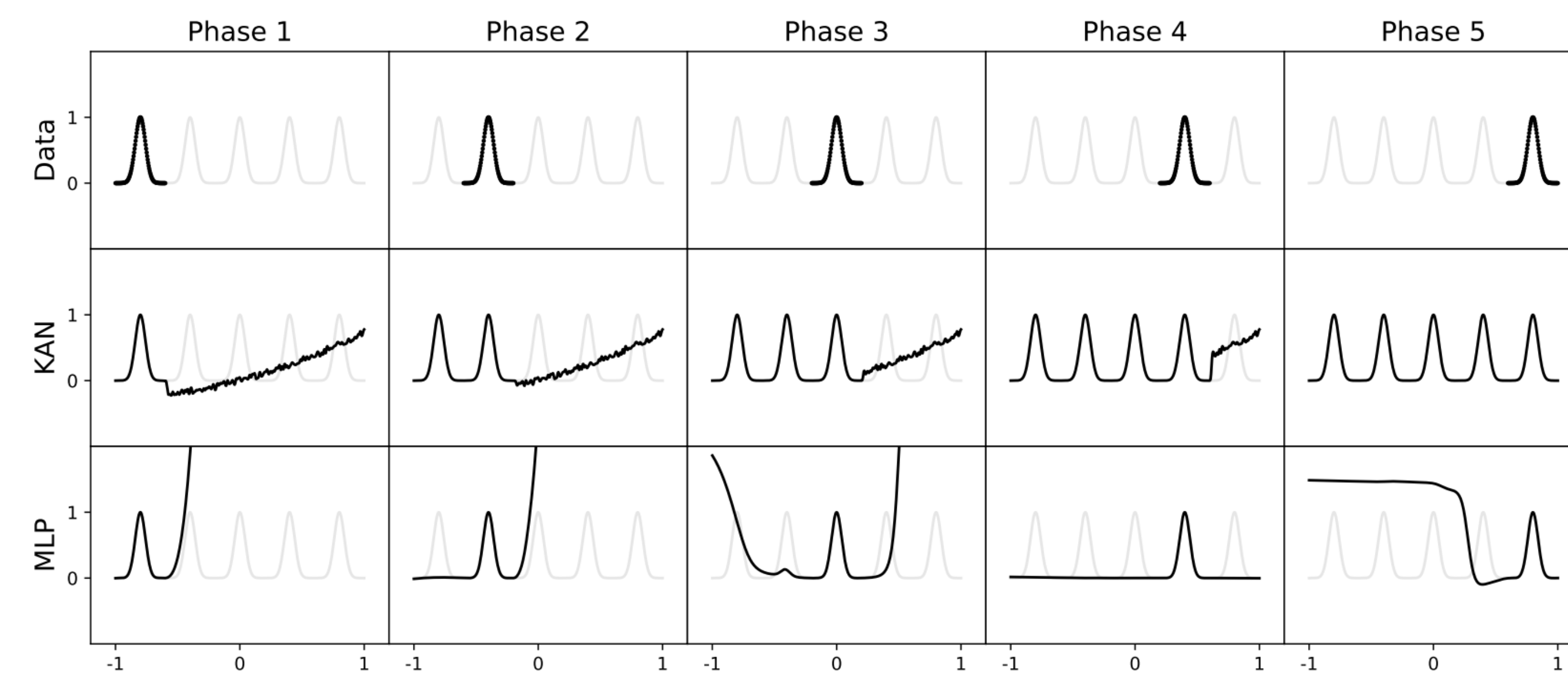


Figura 5. Aprendizaje continuo y falta de memoria. El conjunto de datos es una regresión unidimensional con 5 picos gaussianos (fila superior). Los datos alrededor de cada pico se presentan de forma secuencial (en lugar de todos a la vez) tanto a las KANs como a las MLPs. Las KANs (fila central) evitan completo el olvido catastrófico, mientras que las MLPs (fila inferior) muestran un olvido catastrófico intenso.

Referencias

- [1] Ovidiu Calin. *Deep Learning Architectures A Mathematical Approach* p[296]. Springer Cham, 2020.
- [2] Ziming Liu, Yixuan Wang, Sachin Vaidya, Fabian Ruehle, James Halverson, Marin Soljačić, Thomas Y. Hou, and Max Tegmark. Kan: kolmogorov-arnold networks. *Accepted by International Conference on Learning Representations (ICLR) 2025*, 2024.

Este póster emplea figuras de [2] bajo la licencia CC BY 4.0.