KAN Tutorial Slides

Daniel Precioso Francisco Suárez

October 8, 2025

UNIVERSITY SCHOOL OF SCIENCE & TECHNOLOGY

Kolmogorov-Arnold Representation Theorem

Let $\Omega \subset \mathbb{R}^d$ be a bounded domain and let $f : \Omega \to \mathbb{R}$ be a continuous function; i.e. $f \in C(\Omega)$.
Then there exist continuous univariate functions

$$\Phi_q : \mathbb{R} \to \mathbb{R}, \quad q = 1, \ldots, 2d + 1;$$

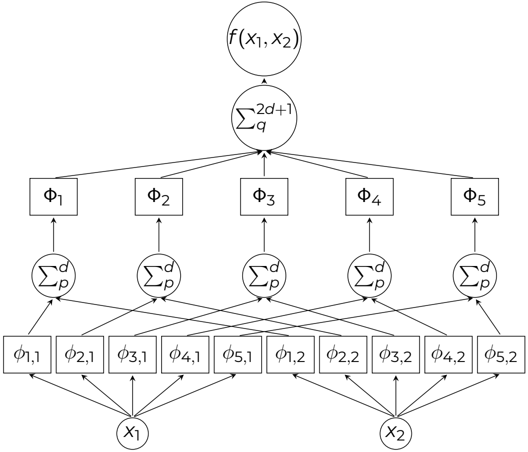
and continuous univariate functions

$$\phi_{pq} : \mathbb{R} \to \mathbb{R}, \quad p = 1, \ldots, d; \quad q = 1, \ldots, 2d + 1;$$

such that for every $\mathbf{x} = (x_1, \ldots, x_d) \in \Omega$,

$$f(\mathbf{x}) = \sum_{q=1}^{2d+1} \Phi_q \left(\sum_{p=1}^{d} \phi_{pq}(x_p) \right).$$

Kolmogorov–Arnold Representation Theorem

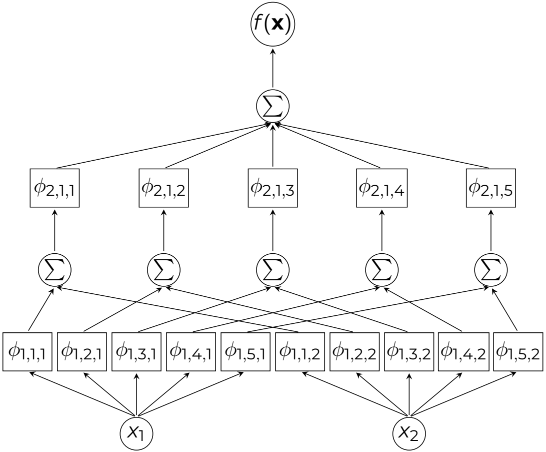


The theorem states that any $f(x_1, x_2)$ can be written as a sum of univariate compositions.

The diagram shows this expression visually: each block represents a component of the decomposition.

Together, they form a **Kolmogorov–Arnold Network (KAN)**.

Kolmogorov–Arnold Networks

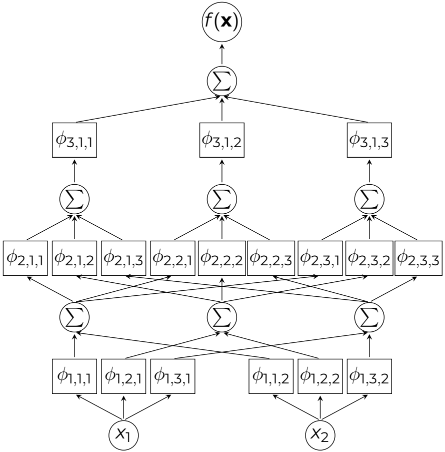


In a network setting, each univariate function is written as $\phi_{d,p,q}$, where:

- d: layer depth
- p: output node index
- q: input node index

This network is a **KAN [2,5,1]**: it has 2 inputs, one hidden layer with 5 nodes, and 1 output.

Kolmogorov–Arnold Networks



KAN [2,3,3,1]: 2 inputs, two hidden layers of 3, 1 output.

Why go deeper?

- **Theory:** Any continuous $f(\mathbf{x})$ admits a shallow KAN $[n_{\text{in}}, 2\,n_{\text{in}}+1, 1]$.
- **Practice:** Deeper KANs can model non-continuous functions. Depth improves expressivity.

B-Splines

$\phi_{d,p,q}$ can be chosen from any family of continuous univariate functions.
A common choice is the **B-spline** family.
A B-spline of degree k is defined as:

$$B_k(x) = \sum_{i=1}^{n} P_i N_{i,k}(x)$$

where n is the number of control points,
$N_{i,k}$ are the basis functions of degree k,
and P_i are the control points (spline weights).

B-Splines

The basis functions follow the standard **Cox–de Boor recursive definition**:

$$N_{i,0}(x) = \begin{cases} 1, & t_i \leq x < t_{i+1} \\ 0, & \text{otherwise} \end{cases}$$

$$N_{i,k}(x) = \frac{x - t_i}{t_{i+k} - t_i} N_{i,k-1}(x) + \frac{t_{i+k+1} - x}{t_{i+k+1} - t_{i+1}} N_{i+1,k-1}(x), \quad k > 0$$

where $t_i \in [t_1, t_m]$ is the **knot vector**, a non-decreasing sequence of real numbers of length $m = n + k + 1$.

B-Splines as KAN Edges

All univariate functions share the same spline degree k and number of control points n. Each $\phi_{d,p,q}$ combines a basis function (similar to residual connections) with a B-spline expansion:

$$\phi(x) = w_b\, b(x) + \sum_{i=1}^{n} P_i\, N_{i,k}(x)$$

Here, w_b is the learnable weight of the basis function, and the control point coefficients P_i scale the individual B-spline functions directly.
We choose the basis as:

$$b(x) = \mathrm{SiLU}(x) = \frac{x}{1 + e^{-x}}$$

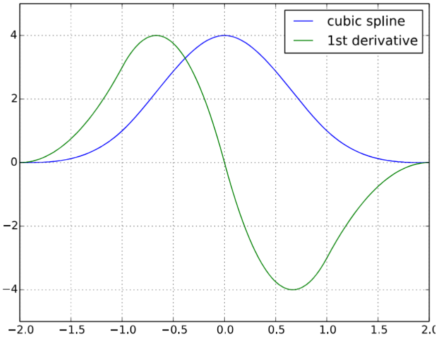
B-Splines as KAN Edges

Knot vector $\mathbf{t} = (t_1, \dots, t_m)$ with

$$m = n+k+1, \qquad a = t_k+1, \quad b = t_{m-k}$$

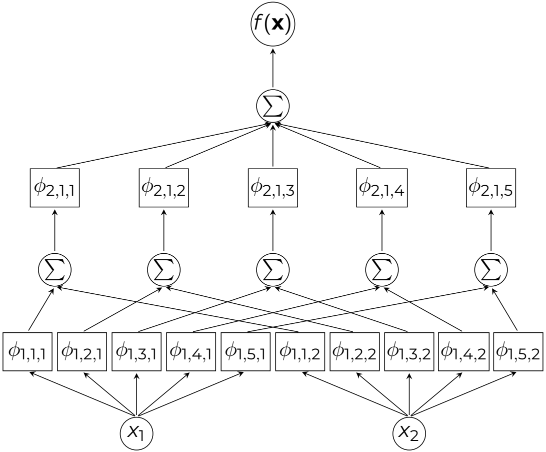
Clamped uniform knots on $[a, b]$:

$$t_1 = \cdots = t_{k+1} = a,$$
$$t_{n+1} = \cdots = t_{n+k+1} = b,$$
$$t_{k+j} = a + \frac{j}{n-k}(b-a), \quad j = 1, \dots, n-k$$



$$\mathbf{t} = (-2, -2, -2, -2, -1, 0, 1, 2, 2, 2, 2)$$
$$\mathbf{P} = (0, 0, 0, 6, 0, 0, 0)$$

KAN Parameters



Hyperparameters

- n: number of control points.
- k: B-spline degree.

Learnable parameters
(for each edge)

- P_i: control points, $i \in [1, n]$.
- w_b: basis weight.

KAN Backpropagation

Loss function is L2 (RMSE):

$$L = \|y - \hat{y}\|_2 = \left\| f(\mathbf{x}) - \hat{f}_d(\mathbf{x}) \right\|_2 = \left\| f(\mathbf{x}) - \sum_q \phi_{d,q}(\mathbf{x}) \right\|_2$$

Where d is the last layer, and $p = 1$ because we have a single output. The coefficients of that layer are $P_{d,q,i}$.

$$\frac{\partial L}{\partial P_{d,q,i}} = \frac{\partial L}{\partial \hat{f}_d(\mathbf{x})} \cdot \frac{\partial \hat{f}_d(\mathbf{x})}{\partial P_{d,q,i}}$$

And for the previous layer $d - 1$:

$$\frac{\partial L}{\partial P_{d-1,p,q,i}} = \frac{\partial L}{\partial \hat{f}_d(\mathbf{x})} \cdot \frac{\partial \hat{f}_d(\mathbf{x})}{\partial \hat{f}_{d-1,p}(\mathbf{x})} \cdot \frac{\partial \hat{f}_{d-1,p}(\mathbf{x})}{\partial P_{d-1,p,q,i}}$$

Capabilities of KANs with B-Splines

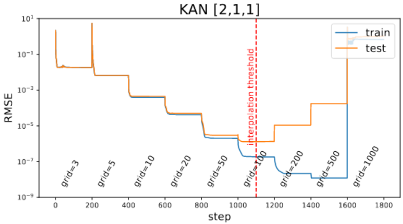
- **Grid extension**: progressively increase model capacity by refining the spline grid without retraining from scratch.
- **Continual learning**: local support ensures new information affects only nearby regions, reducing catastrophic forgetting.
- **Sparsity**: regularization and pruning remove redundant components, simplifying the model without major accuracy loss.
- **Symbolic regression**: univariate structure enables conversion of learned functions into interpretable closed-form expressions.

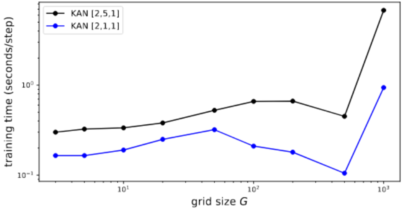
Capabilities - Grid Extension

Grid extension refines a trained KAN by adding more spline knots without restarting training.

- Train on a coarse grid first.
- Add knots to increase resolution and capacity.
- Initialize new coefficients by least-squares fitting.
- Continue training to improve accuracy.

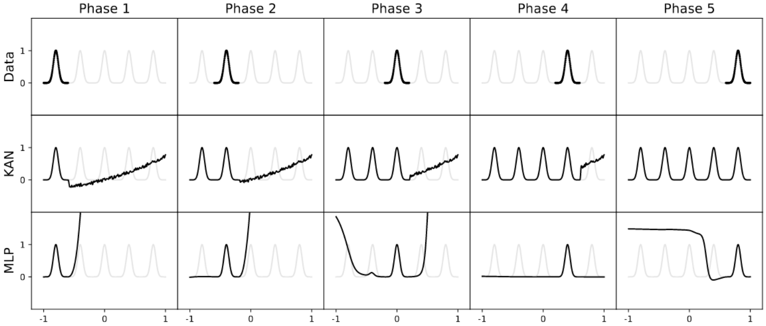
Test loss often improves until the parameter count roughly matches the number of data points.





Capabilities - Continual Learning

Because B-splines have **local support**, updates to $\phi(x)$ in one region of the input space affect only nearby points. This locality mitigates **catastrophic forgetting**, a common issue in MLPs where learning new data can overwrite previously acquired knowledge.

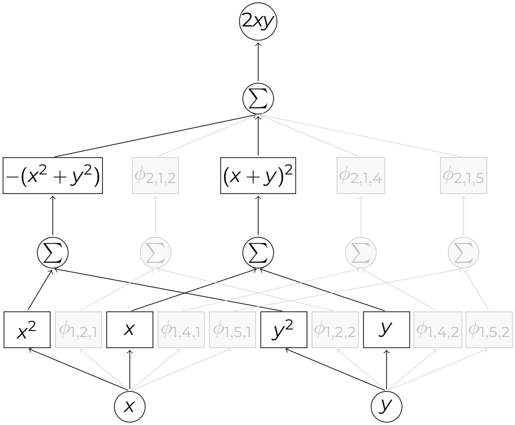


Capabilities - Sparsity

Sparsity removes unnecessary components, revealing the essential structure of our target function.

- **Regularization** drives many spline weights toward zero.
- Irrelevant $\phi_{d,p,q}$ can be pruned after training.
- The result is a compact, interpretable network.

Sparsity helps towards **interpretability**.



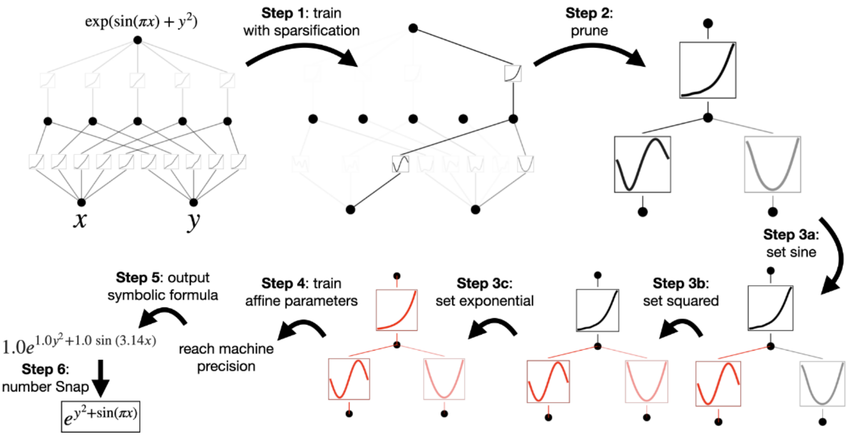
Capabilities - Symbolic Regression

KANs provide an interpretable path from neural models to closed-form expressions:

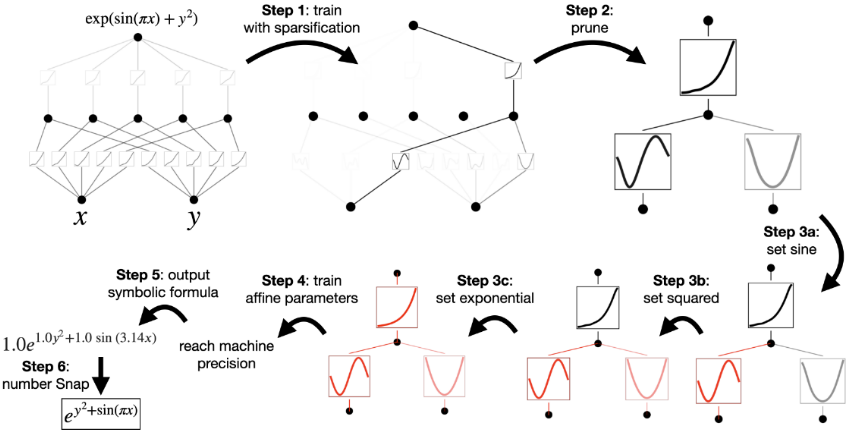
- Each learned $\phi_{d,p,q}$ is a univariate function, which can often be approximated by simple analytic forms (e.g., $\sin$, $\exp$, $\log$).
- After training, these functions are "snapped" to symbolic templates via affine fitting, producing human-readable equations.
- The resulting network can be viewed as a composition graph of symbolic functions approximating $f(x)$.

This makes KANs suitable not only for prediction but also for **discovering interpretable laws** from data.

KAN Train Steps



Limitations of KANs with B-Splines



References