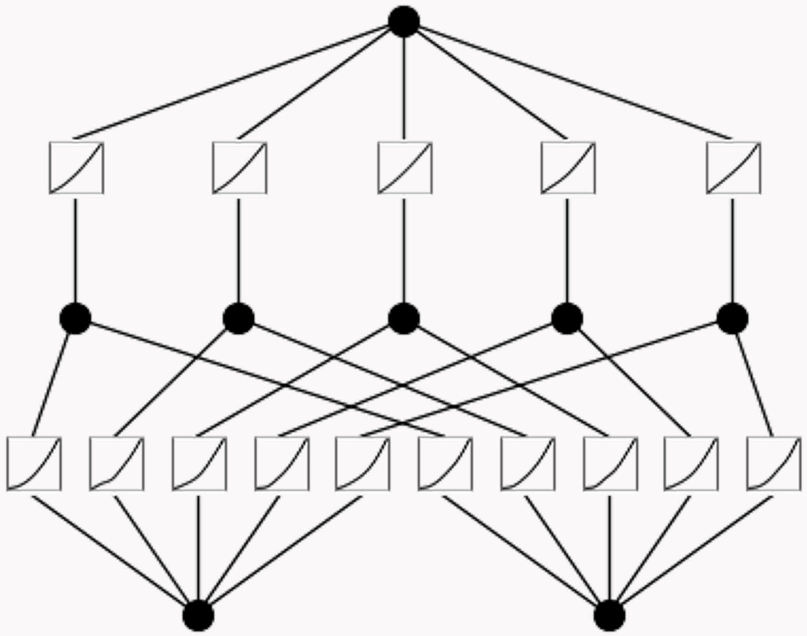
Kolmogorov-Arnold Network (KAN)

Keywords
Promising alternatives to MLPs?
Better accuracy with fewer parameters
More interpretable
Learnable activation functions
Can avoid catastrophic forgetting
Slower and harder to train



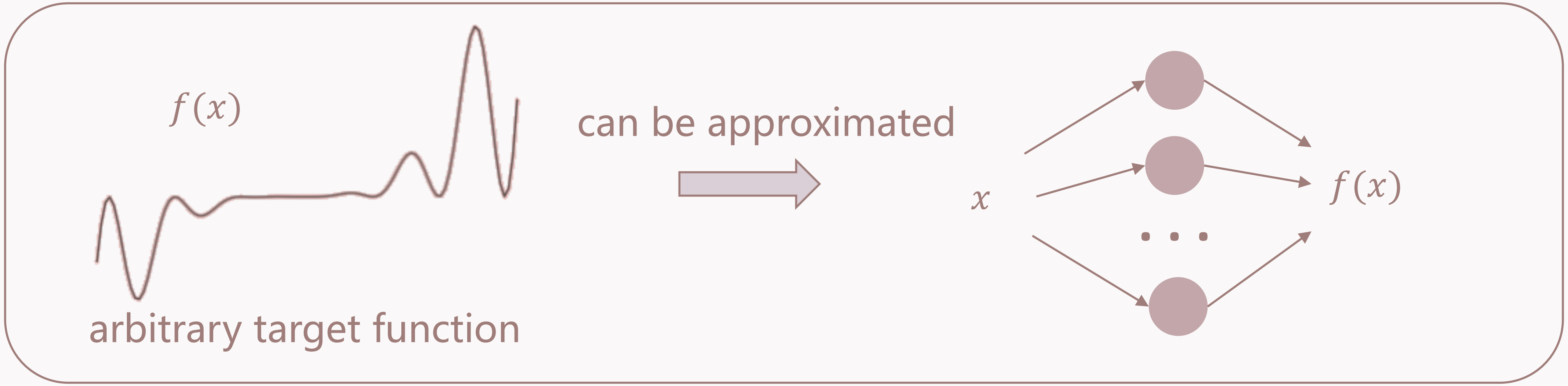
Universal Approximation Theorem

Let $C(X, \mathbb{R}^m)$ denote the set of continuous functions from a subset X of a Euclidean $\mathbb{R}^n$ space to a Euclidean space $\mathbb{R}^m$. Let $\sigma \in C(\mathbb{R}, \mathbb{R})$. Note that $(\sigma \circ x)_i = \sigma(x_i)$,so $\sigma \circ x$ denotes σ applied to each component of x.

Then σ is not polynomial if and only if for every $n \in \mathbb{N}, m \in \mathbb{N}$, compact $K \subseteq \mathbb{R}^n$, $\mathrm{f} \in C(K, \mathbb{R}^m)$,$\varepsilon > 0$,there exist $\mathrm{k} \in \mathbb{N}, W \in \mathbb{R}^{k \times n}$, $b \in \mathbb{R}^k$, $C \in \mathbb{R}^{k \times n}$,such that:

$$\sup_{x \in K} \| f(x) - g(x) \| < \varepsilon$$

where $g(x) = C \cdot (\sigma \circ (W \cdot x + b))$



Universal Approximation Theorem

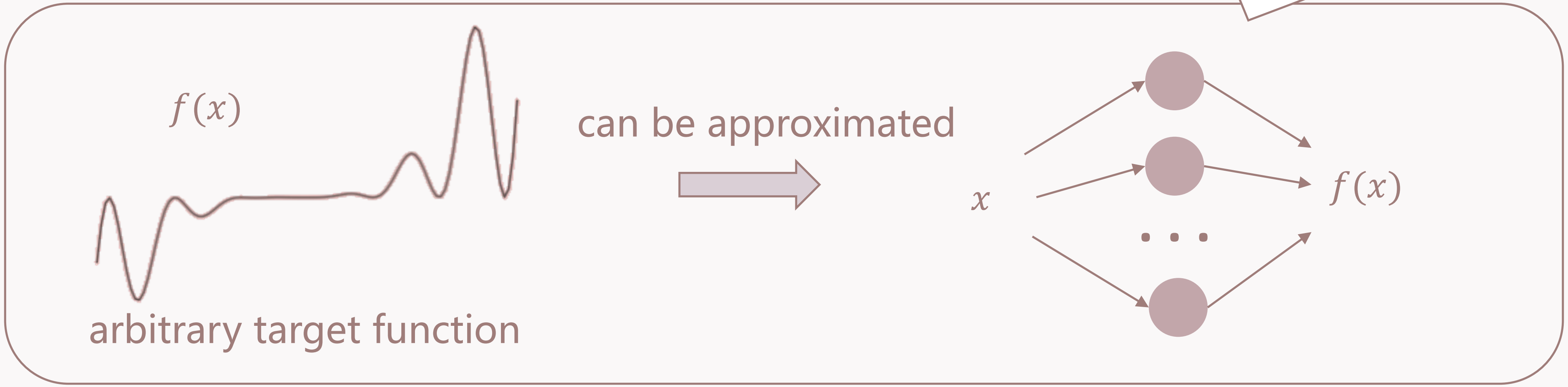
Let $C(X, \mathbb{R}^m)$ denote the set of continuous functions from a subset X of a Euclidean $\mathbb{R}^n$ space to a Euclidean space $\mathbb{R}^m$. Let $\sigma \in C(\mathbb{R}, \mathbb{R})$. Note that $(\sigma \circ x)_i = \sigma(x_i)$,so $\sigma \circ x$ denotes σ applied to each component of x.

Then σ is not polynomial if and only if for every $n \in \mathbb{N}, m \in \mathbb{N}$, compact $K \subseteq \mathbb{R}^n$, $\mathrm{f} \in C(K, \mathbb{R}^m)$,$\varepsilon > 0$,there exist $\mathrm{k} \in \mathbb{N}, W \in \mathbb{R}^{k\times n}$, $b \in \mathbb{R}^k$, $C \in \mathbb{R}^{k\times n}$,such that:

$$\sup_{x\in K} \| f(x) - g(x) \| < \varepsilon$$

where $g(x) = C \cdot (\sigma \circ (W \cdot x + b))$

Drawbacks:
lack interpretable,
too many parameters



Kolmogorov-Arnold Representation Theorem

If f is a multivariate continuous function, then f can be written as a finite composition of continuous functions of a single variable and the binary operation of addition. More specifically,

$$f(x) = f(x_1, \dots, x_n) = \sum_{q=1}^{2n+1} \Phi_q \left(\sum_{p=1}^{n} \phi_{q,p}(x_p) \right)$$

Where $\phi_{q,p}: [0,1] \to \mathbb{R}$ and $\Phi_q: \mathbb{R} \to \mathbb{R}$.

e.g.

$$f(x, y) = xy = \underbrace{\exp(\log(x+1) + \log(y+1))} - \overbrace{(x+0.5)}^{\text{univariate}} - \overbrace{(y+0.5)}^{\text{univariate}}$$

Univariate, with variable of $\underbrace{\log(x+1)}_{\text{univariate}} + \underbrace{\log(y+1)}_{\text{univariate}}$

Kolmogorov-Arnold Representation Theorem

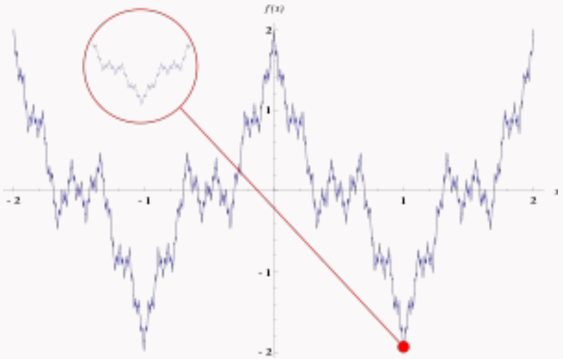
If f is a multivariate continuous function, then f can be written as a finite composition of continuous functions of a single variable and the binary operation of addition. More specifically,

$$f(x) = f(x_1, \dots, x_n) = \sum_{q=1}^{2n+1} \Phi_q\left(\sum_{p=1}^{n} \phi_{q,p}(x_p)\right)$$

Where $\phi_{q,p}: [0,1] \to \mathbb{R}$ and $\Phi_q: \mathbb{R} \to \mathbb{R}$.

can be

non-smooth



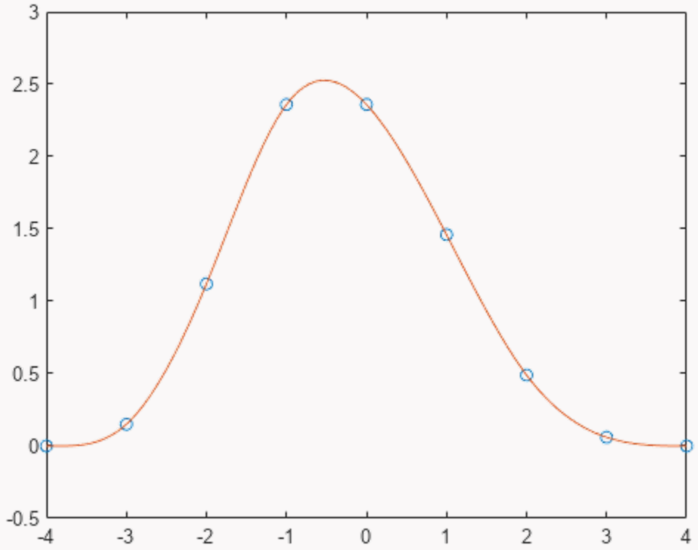
fractal

Not learnable in practice!

Spline

A spline function S is said to be of order k that greater than or equal to one on the interval $a = t_0 < t_1 < \cdots < t_n = b$, if it satisfies the following two properties:

1. S is a polynomial of degree that is less than k on each of the subintervals $[t_i, t_{i+1}]$.
2. The derivative of the spine function is continuous on the full interval $[a, b]$ for all the derivatives up to $k - 1$.



Advantages: Performs well in data interpolation and function approximation in low dimensional space

Disadvantage: Curse of Dimensionality

Spline

A spline function S is said to be of order k that greater than or equal to one on the interval $a = t_0 < t_1 < \cdots < t_n = b$, if it satisfies the following two properties:

1. S is a polynomial of degree that is less than k on each of the subintervals $[t_i, t_{i+1}]$.
2. The derivative of the spine function is continuous on the full interval $[a, b]$ for all the derivatives up to $k - 1$.

The B-Splines can be defined through the Cox-de Boor recursive formula :

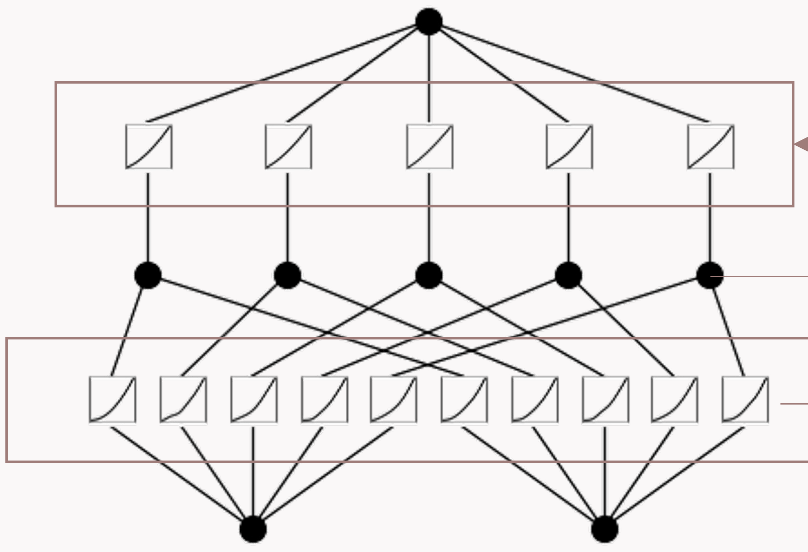
$$B_{i,0}(x) := \begin{cases} 1 & if\ t_i \le x < t_{i+1}, \\ 0 & otherwise. \end{cases}$$

$$B_{i,k}(x) := \frac{x - t_i}{t_{i+k} - t_i} B_{i,k-1}(x) + \frac{t_{i+k+1} - x}{t_{i+k+1} - t_{i+1}} B_{i+1,k-1}(x).$$

KAN Architecture

l	index of the layer
i	index of x
p	index of the input dimension n_in
q	index of the output dimension n_out
n	input dimension

Output: $KAN(x) = (\Phi_{L-1} \circ \Phi_{L-2} \ldots \circ \Phi_1 \circ \Phi_0)x$



Input dimension: $n_{in} = 2\text{n} + 1$
Output dimension: $n_{out} = 1$

Sum operation on nodes

Learnable activation function$\phi_{l,p,q}$

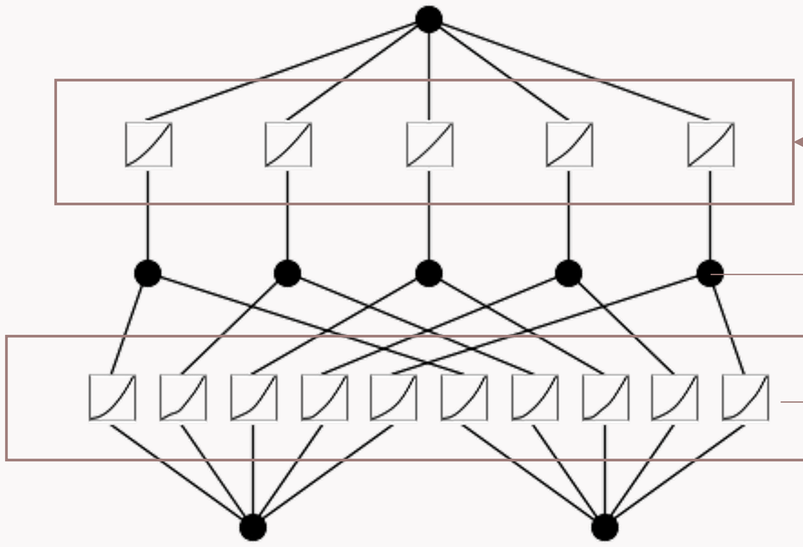
Input: $x_{l,i}$

A KAN layer: $\Phi_l = \{\phi_{p,q}\} = \begin{bmatrix} \phi_{l,1,1}(\cdot) & \phi_{l,1,2}(\cdot) & \cdots & \phi_{l,1,n_l}(\cdot) \\ \phi_{l,2,1}(\cdot) & \phi_{l,2,2}(\cdot) & \cdots & \phi_{l,2,n_l}(\cdot) \\ \vdots & \vdots & \ddots & \vdots \\ \phi_{l,n_{l+1},1}(\cdot) & \phi_{l,n_{l+1},2}(\cdot) & \cdots & \phi_{l,n_{l+1},n}(\cdot) \end{bmatrix}$

Input dimension: $n_{in} = \text{n}$
Output dimension: $n_{out} = 2\text{n} + 1$

An Introduction to Kolmogorov-Arnold Networks

Output: $KAN(x) = (\Phi_{L-1} \circ \Phi_{L-2} \ldots \circ \Phi_1 \circ \Phi_0)x$



Input dimension: $n_{in} = 2\mathrm{n} + 1$
Output dimension: $n_{out} = 1$

Sum operation on nodes

Learnable activation function$\phi_{l,p,q}$

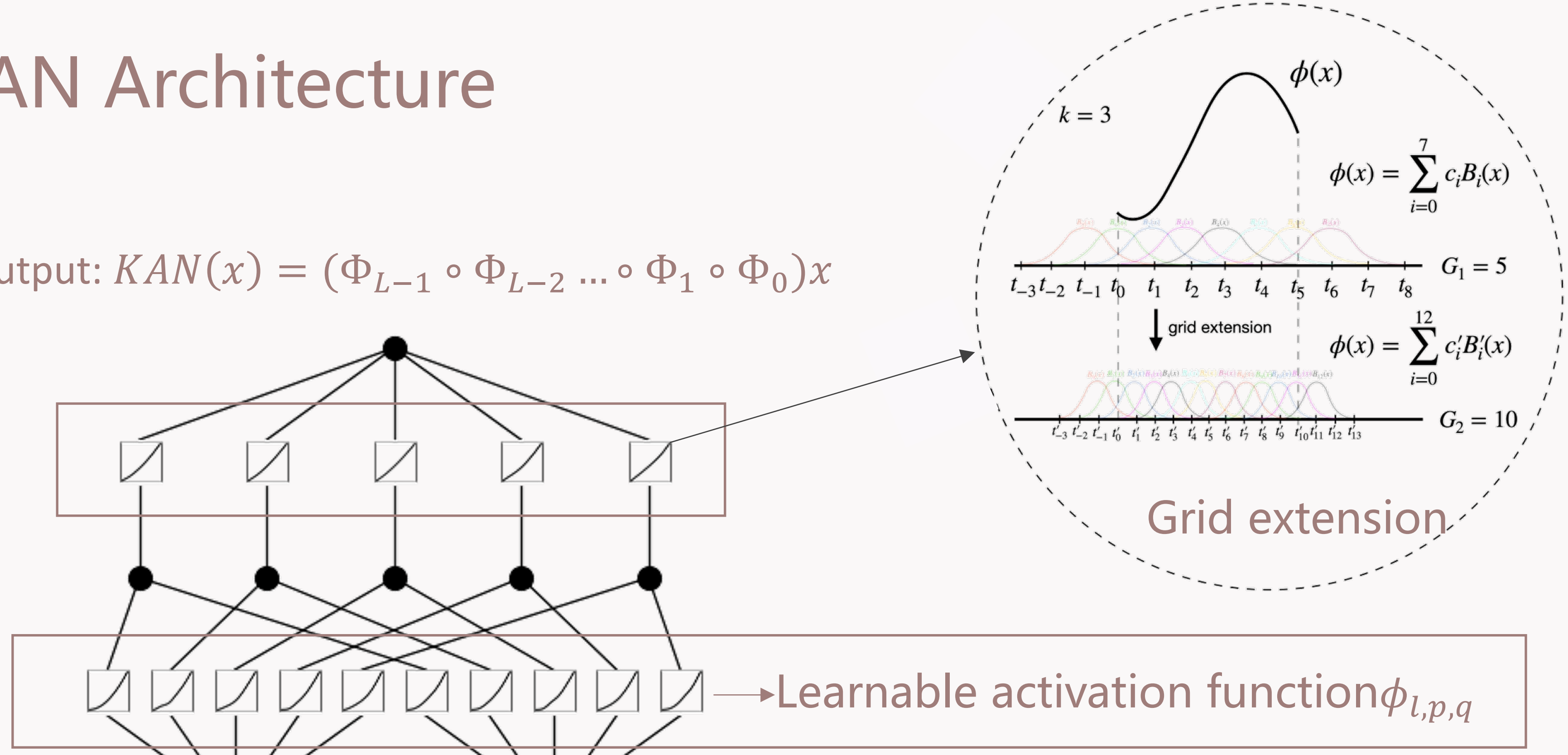
Input: $x_{l,i}$

A KAN layer: $\Phi_l = \{\phi_{p,q}\} = \begin{bmatrix} \phi_{l,1,1}(\cdot) & \phi_{l,1,2}(\cdot) & \cdots & \phi_{l,1,n_l}(\cdot) \\ \phi_{l,2,1}(\cdot) & \phi_{l,2,2}(\cdot) & \cdots & \phi_{l,2,n_l}(\cdot) \\ \vdots & \vdots & \ddots & \vdots \\ \phi_{l,n_{l+1},1}(\cdot) & \phi_{l,n_{l+1},2}(\cdot) & \cdots & \phi_{l,n_{l+1},n}(\cdot) \end{bmatrix}$

Input dimension: $n_{in} = \mathrm{n}$
Output dimension: $n_{out} = 2\mathrm{n} + 1$

KAN Architecture

Output: $KAN(x) = (\Phi_{L-1} \circ \Phi_{L-2} \ldots \circ \Phi_1 \circ \Phi_0)x$


$k = 3$
$\phi(x)$
$\phi(x) = \sum_{i=0}^{7} c_i B_i(x)$
$G_1 = 5$
$t_{-3}\ t_{-2}\ t_{-1}\ t_0\ t_1\ t_2\ t_3\ t_4\ t_5\ t_6\ t_7\ t_8$
grid extension
$\phi(x) = \sum_{i=0}^{12} c_i' B_i'(x)$
$G_2 = 10$
$t'_{-3}\ t'_{-2}\ t'_{-1}\ t'_0\ t'_1\ t'_2\ t'_3\ t'_4\ t'_5\ t'_6\ t'_7\ t'_8\ t'_9\ t'_{10}\ t'_{11}\ t'_{12}\ t'_{13}$

Grid extension

Learnable activation function$\phi_{l,p,q}$

w is initialized according to Xavier initialization

Input: $x_{l,i}$

Residual activation functions：

$$\phi(x) = w(b(x) + spline(x))$$

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

$$spline(x) = \sum_i c_i B_i(x)$$

$spline(x)$ is initialized to $\approx 0^2$

Interpretability of KAN

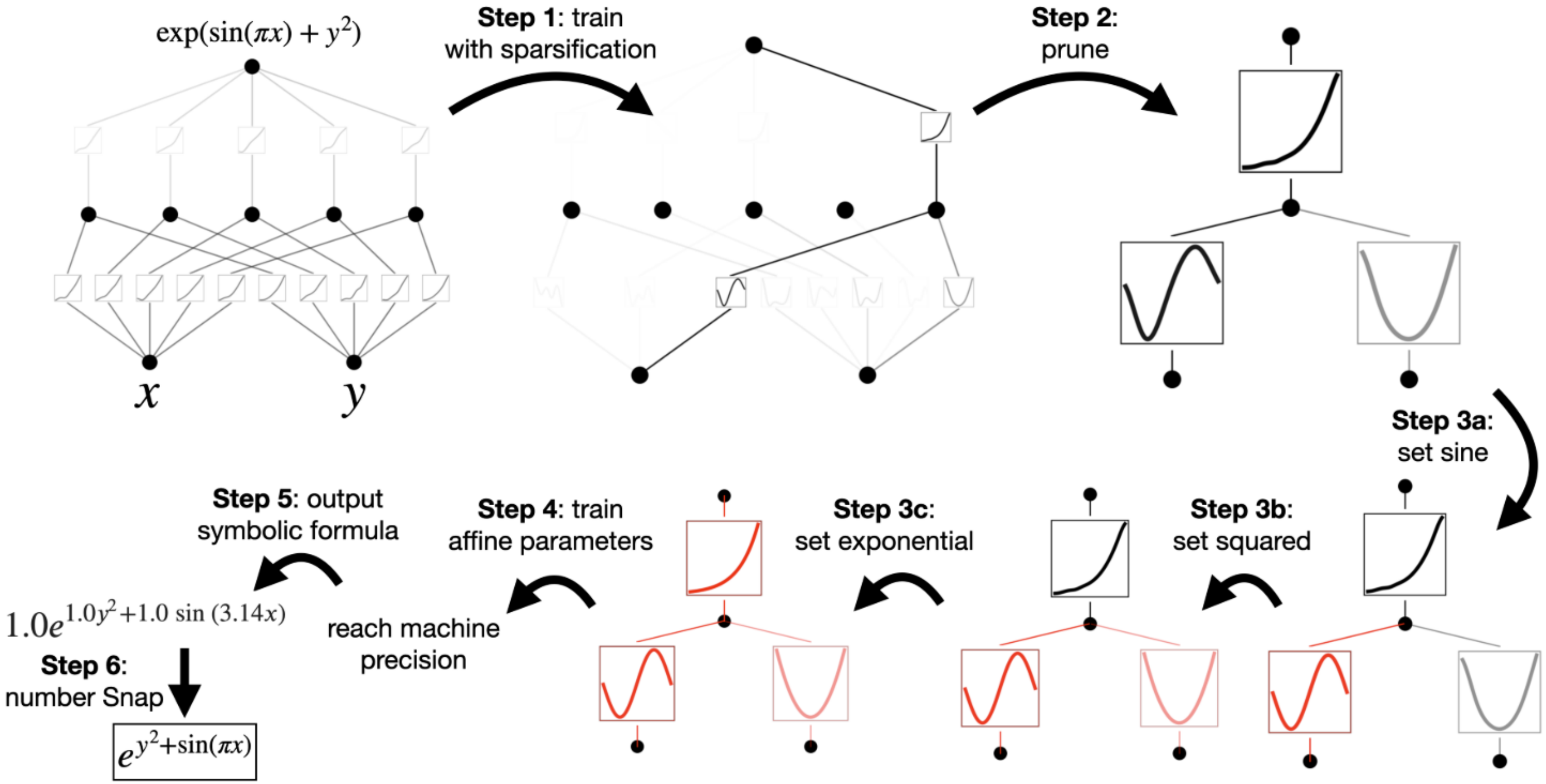

$\exp(\sin(\pi x) + y^2)$
x
y
Step 1: train with sparsification
Step 2: prune
Step 3a: set sine
Step 3b: set squared
Step 3c: set exponential
Step 4: train affine parameters
reach machine precision
Step 5: output symbolic formula
$1.0e^{1.0y^2+1.0\sin(3.14x)}$
Step 6: number Snap
$\boxed{e^{y^2+\sin(\pi x)}}$

Figure 2.4: An example of how to do symbolic regression with KAN.

Fitting Special Functions

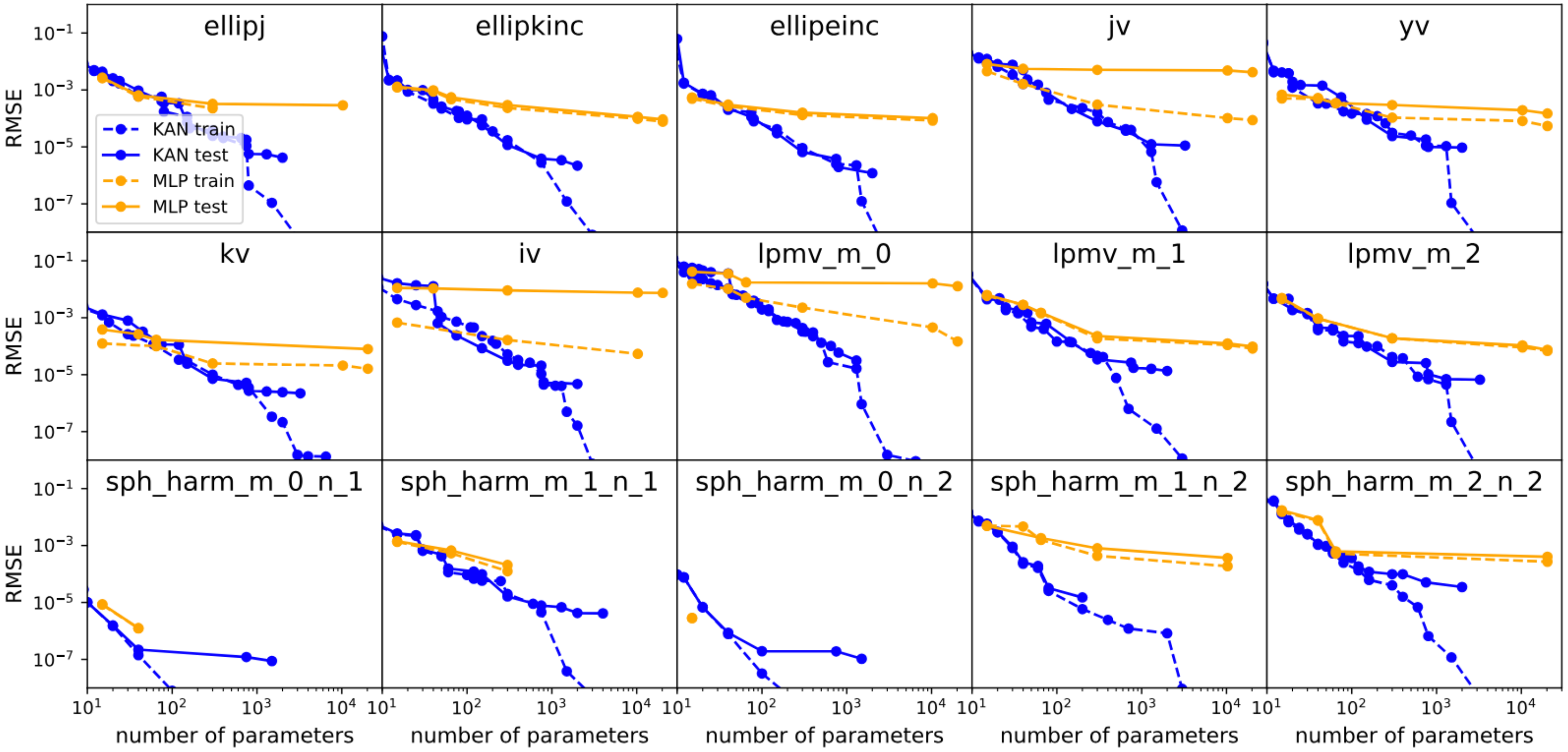


Figure 3.2: Fitting special functions. We show the Pareto Frontier of KANs and MLPs in the plane spanned by the number of model parameters and RMSE loss. Consistently accross all special functions, KANs have better Pareto Frontiers than MLPs. The definitions of these special functions are in Table 2.

Solving Partial Differential Equations

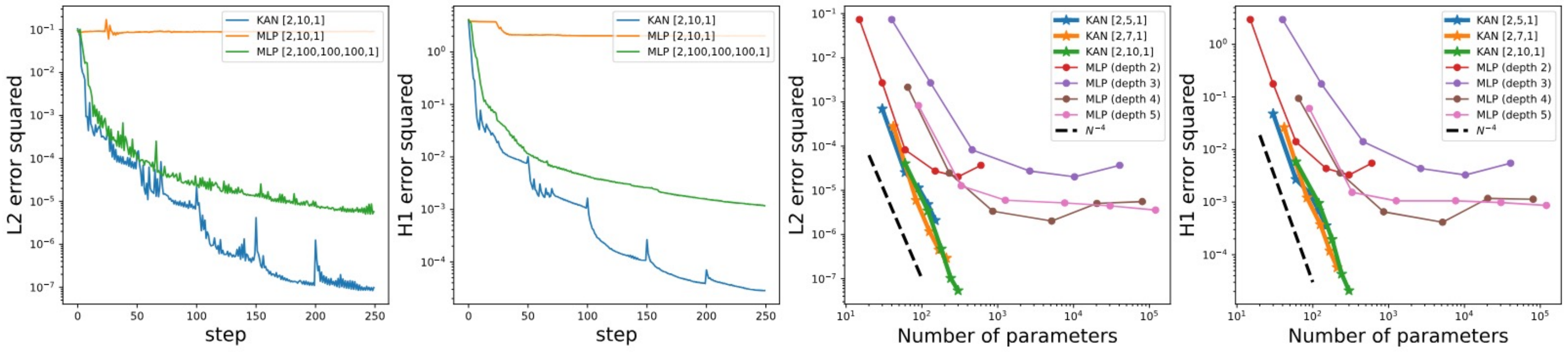


Figure 3.3: The PDE example. We plot L2 squared and H1 squared losses between the predicted solution and ground truth solution. First and second: training dynamics of losses. Third and fourth: scaling laws of losses against the number of parameters. KANs converge faster, achieve lower losses, and have steeper scaling laws than MLPs.

Continue Learning

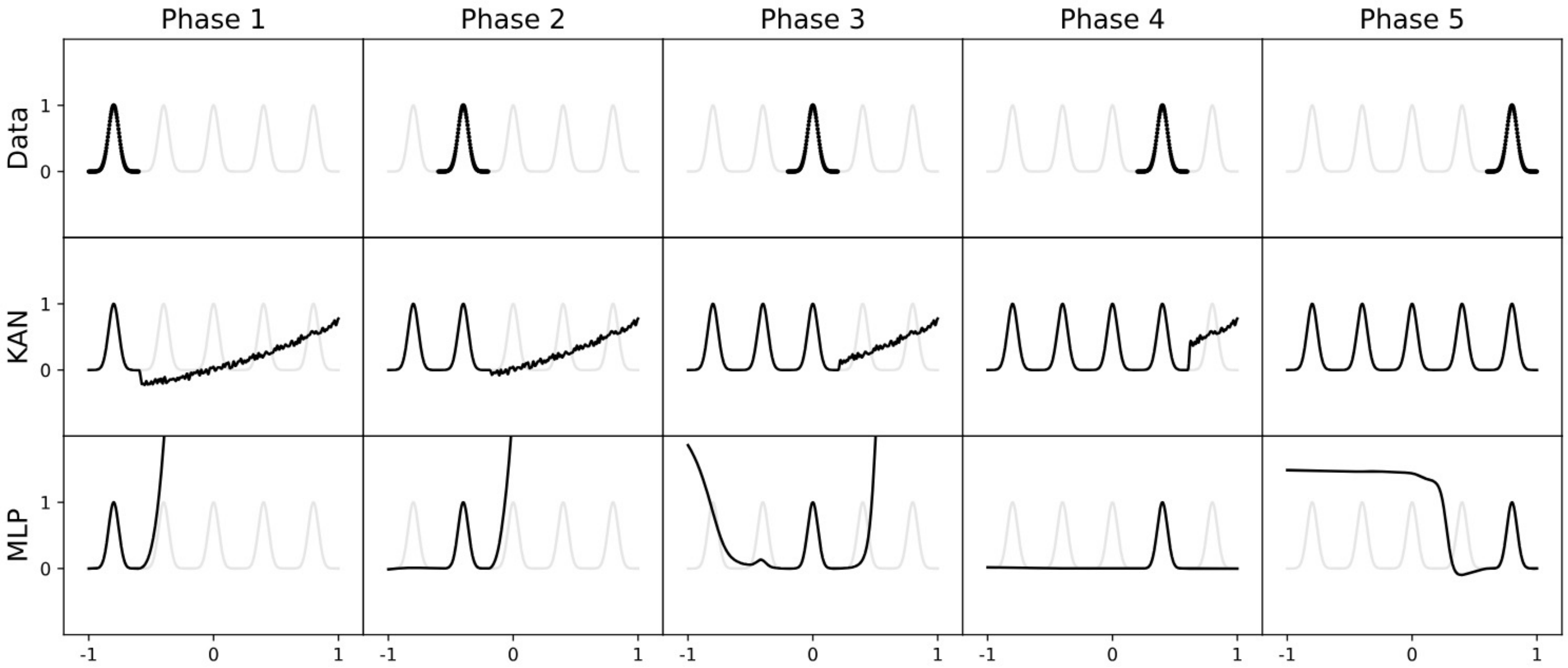


Figure 3.4: A toy continual learning problem. The dataset is a 1D regression task with 5 Gaussian peaks (top row). Data around each peak is presented sequentially (instead of all at once) to KANs and MLPs. KANs (middle row) can perfectly avoid catastrophic forgetting, while MLPs (bottom row) display severe catastrophic forgetting.

MLPs or KANs?

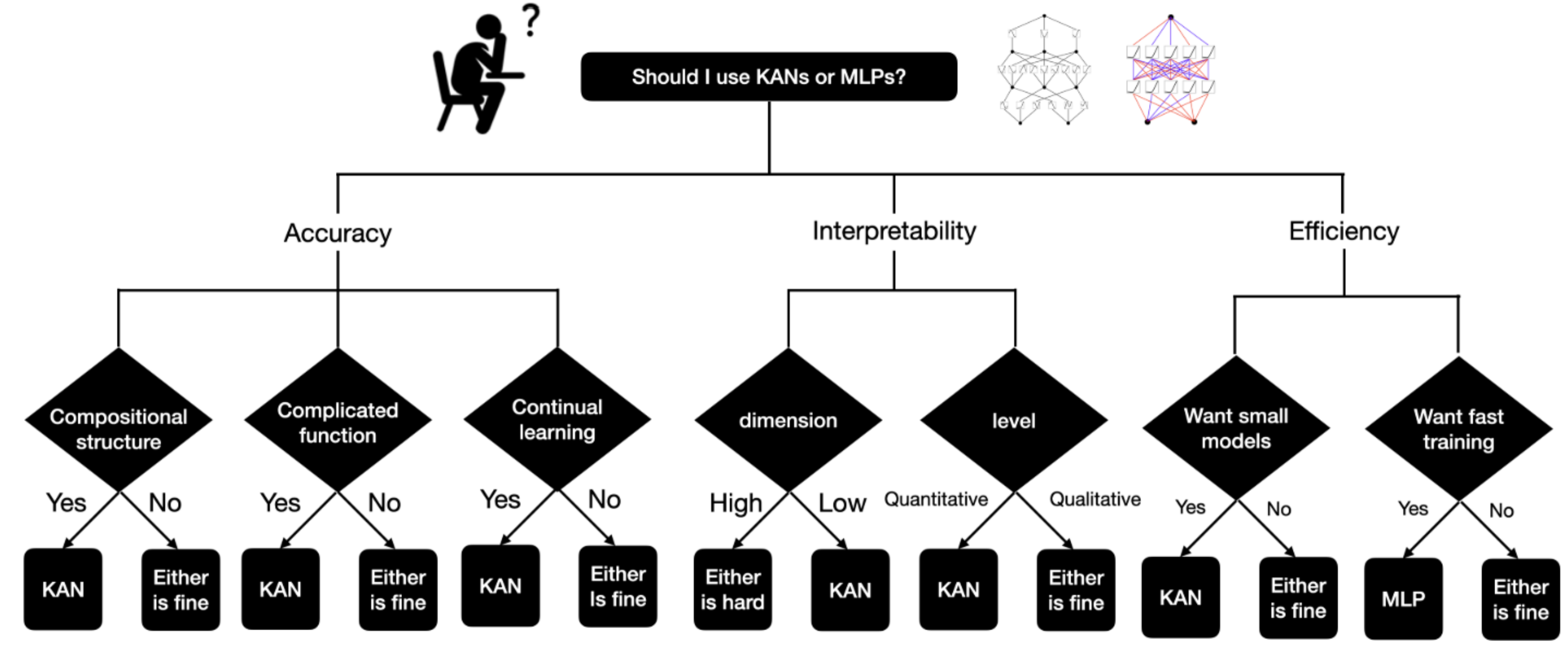


Figure 6.1: Should I use KANs or MLPs?