Kolmogorov-Arnold Networks

Mayank Singh

Multi Layer Perceptrons

- Building blocks of modern AI
- Expressive power guaranteed by Universal Approximation Theorem
- Drawbacks:
    - Less interpretable
    - Almost all non-embedding parameters
- Can we do better?

Kolmogorov-Arnold Networks

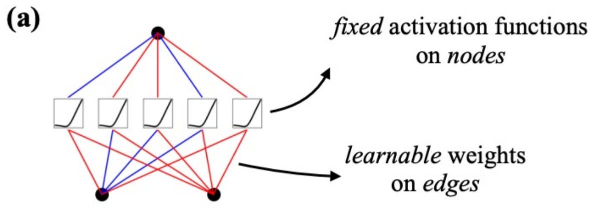
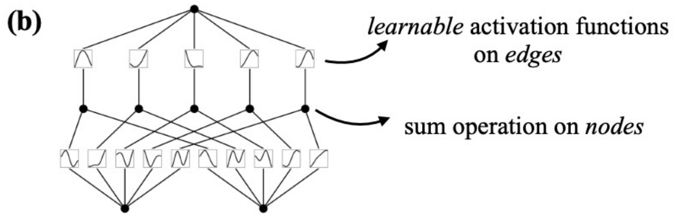
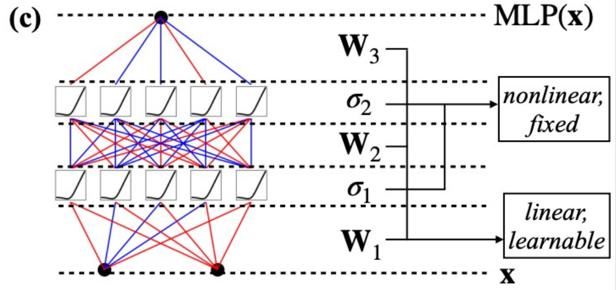
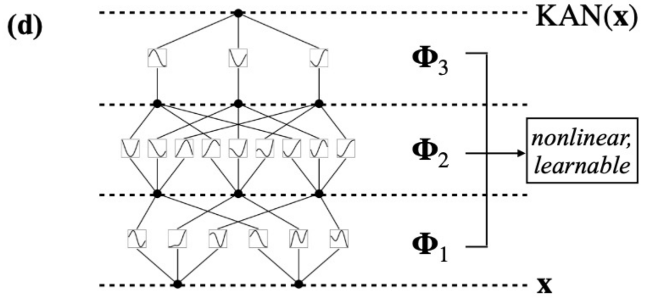
- Inspired by Kolmogorov Arnold Theorem
- Replace activations with splines (1D)
- No edge weights to learn; only spline weights
- Benefits:
  - Accurate
  - Interpretable

Kolmogorov-Arnold Networks

$$f(x_1, \cdots, x_N) = \exp\left(\frac{1}{N}\sum_{i=1}^{N} \sin^2(x_i)\right)$$

Splines – Fail due to curse of dimensionality
MLPs – ReLUs cannot express the sines and exponentials accurately
KANs – Can discover the formula from data

MLP vs KAN

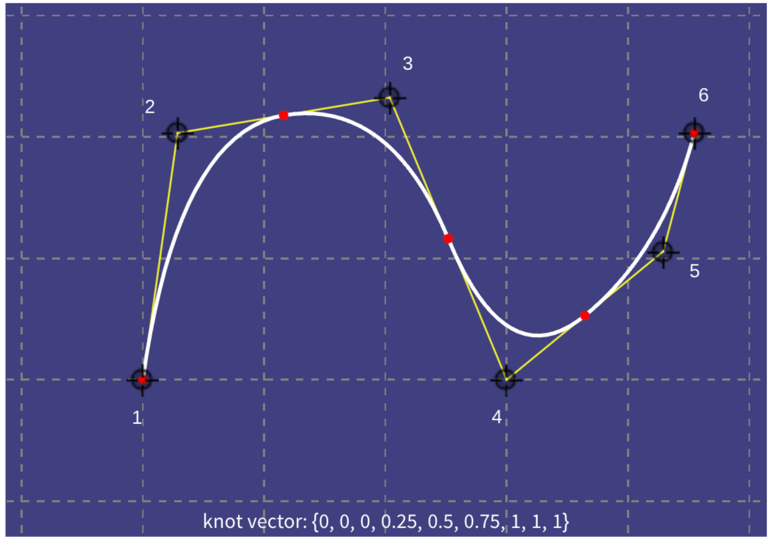
<table>
<tr><th>Model</th><th>Multi-Layer Perceptron (MLP)</th><th>Kolmogorov-Arnold Network (KAN)</th></tr>
<tr><td>Theorem</td><td>Universal Approximation Theorem</td><td>Kolmogorov-Arnold Representation Theorem</td></tr>
<tr><td>Formula (Shallow)</td><td>$f(\mathbf{x}) \approx \sum_{i=1}^{N(\epsilon)} a_i \sigma(\mathbf{w}_i \cdot \mathbf{x} + b_i)$</td><td>$f(\mathbf{x}) = \sum_{q=1}^{2n+1} \Phi_q \left(\sum_{p=1}^{n} \phi_{q,p}(x_p) \right)$</td></tr>
<tr><td>Model (Shallow)</td><td>
</td><td>
</td></tr>
<tr><td>Formula (Deep)</td><td>$\mathrm{MLP}(\mathbf{x}) = (\mathbf{W}_3 \circ \sigma_2 \circ \mathbf{W}_2 \circ \sigma_1 \circ \mathbf{W}_1)(\mathbf{x})$</td><td>$\mathrm{KAN}(\mathbf{x}) = (\mathbf{\Phi}_3 \circ \mathbf{\Phi}_2 \circ \mathbf{\Phi}_1)(\mathbf{x})$</td></tr>
<tr><td>Model (Deep)</td><td>
</td><td>
</td></tr>
</table>

Activations in KANs

- Activation functions are more heavy-duty and parametrized
- Smooth curves using Splines
- Why smooth?
  - For differentiability and end-to-end learning

Splines

- Composed of piece-wise basis functions (B-Splines)
- Express “almost any” smooth curve
- Why splines?
  - Can be made arbitrarily accurate through number of basis functions (grid size)
  - More accurate for low-dimensional curves than MLPs
  - Struggle at higher-dimensions



$$\mathrm{spline}(x) = \sum_i c_i B_i(x)$$

Universal Approximation Theorem

- Feedforward neural network
  - with a single hidden layer
  - containing a finite number of neurons
  - can approximate any continuous function
  - (Some constraints)

Kolmogorov-Arnold Representation Theorem

for a smooth $f : [0,1]^n \to \mathbb{R}$

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

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

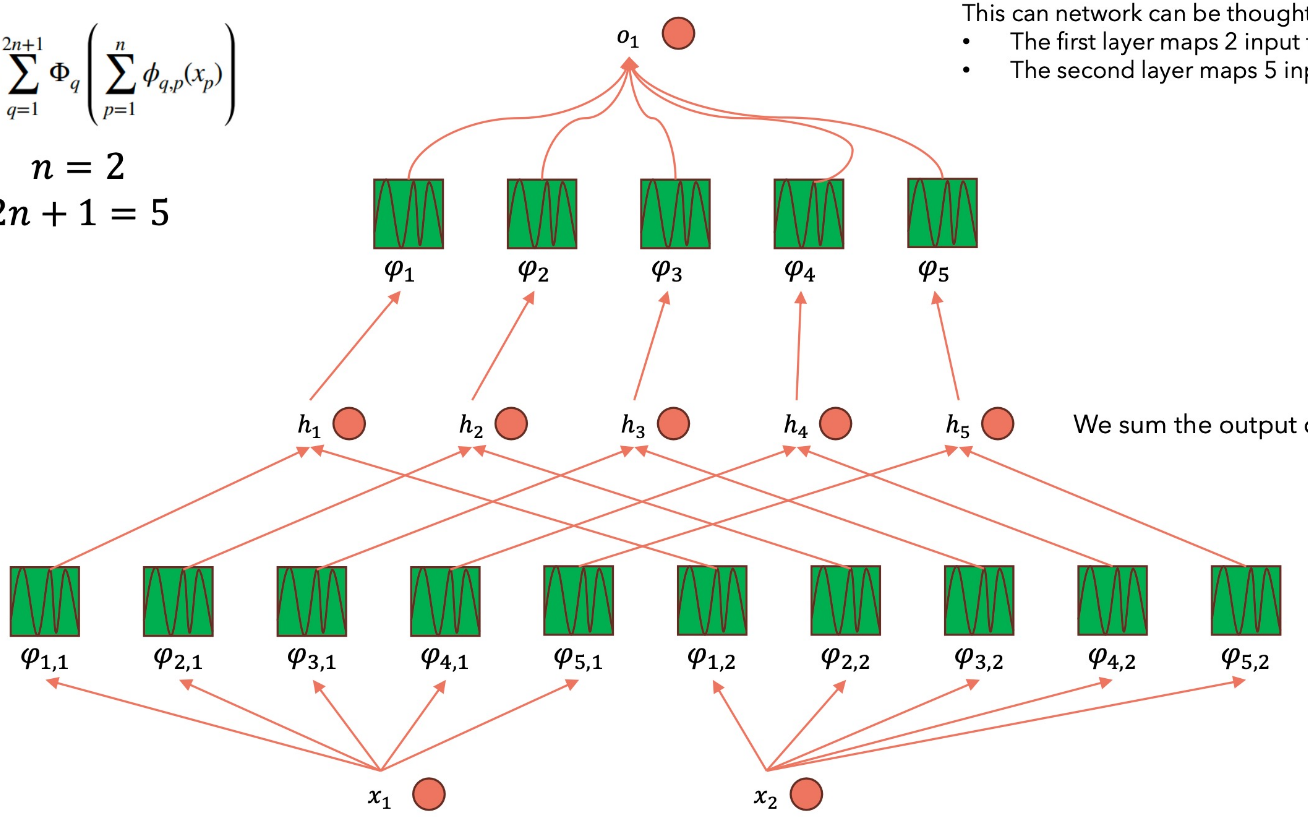
Kolmogorov-Arnold Representation Theorem

- f = Multivariate continuous function on a bounded domain
- Can be expressed as:
  - Finite composition of continuous functions of a single variable and addition
- Only true multivariate function is addition
- 1D functions can be non-smooth and even fractal (not learnable)

Kolmogorov-Arnold Networks

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

$$n = 2$$
$$2n + 1 = 5$$



This can network can be thought of as two layers applied in sequence:

- The first layer maps 2 input features into 5 output features.
- The second layer maps 5 input features into 1 output feature.

We sum the output of the learnable functions

Instead of having learnable weights, we have **learnable functions**

Adapted from Umar Jamil's slides on KANs

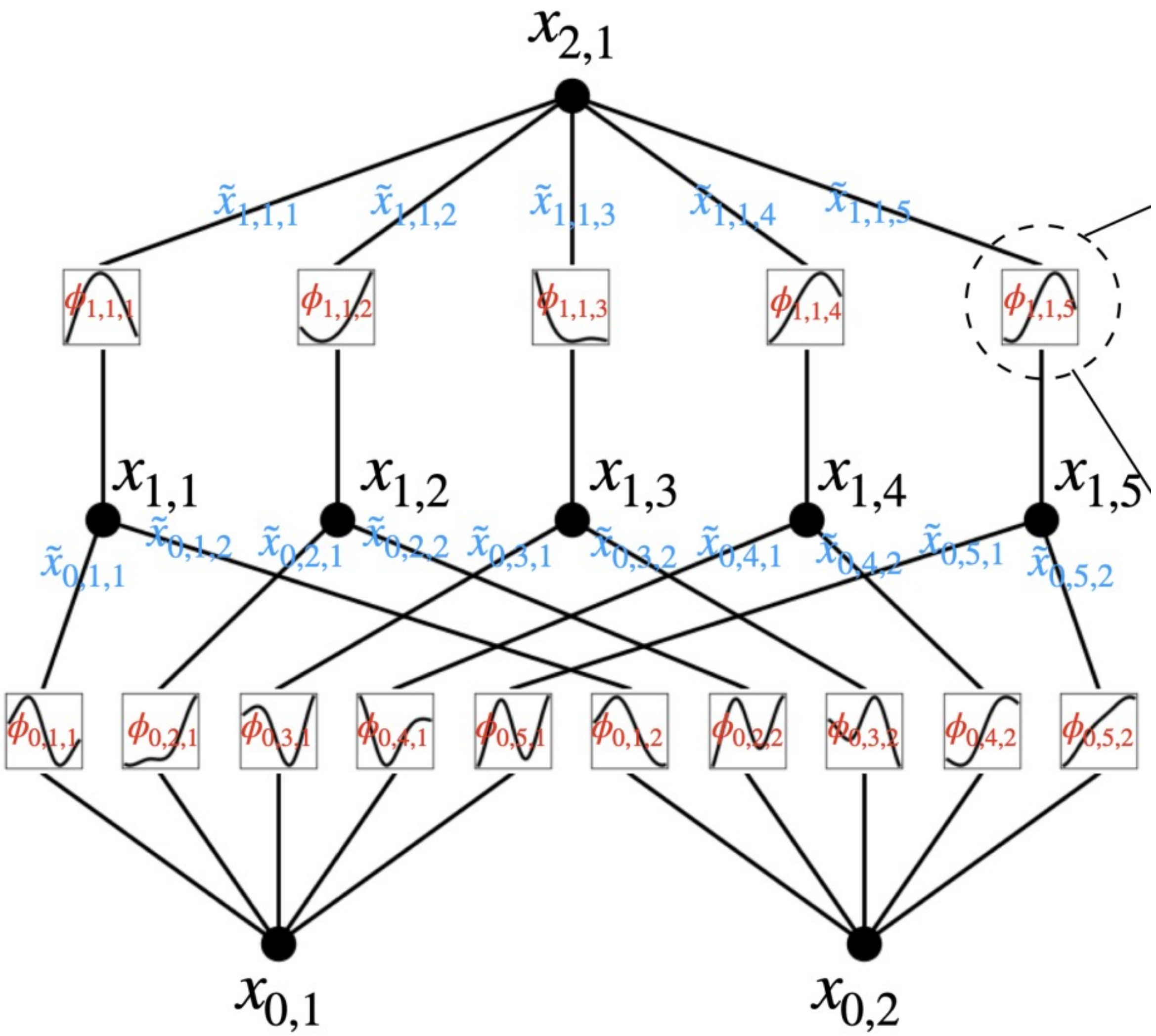
Deep KANs

$$\phi_{l,j,i}, \quad l = 0, \cdots, L-1, \quad i = 1, \cdots, n_l, \quad j = 1, \cdots, n_{l+1}.$$

l = Layer number
i = input features
j = output features

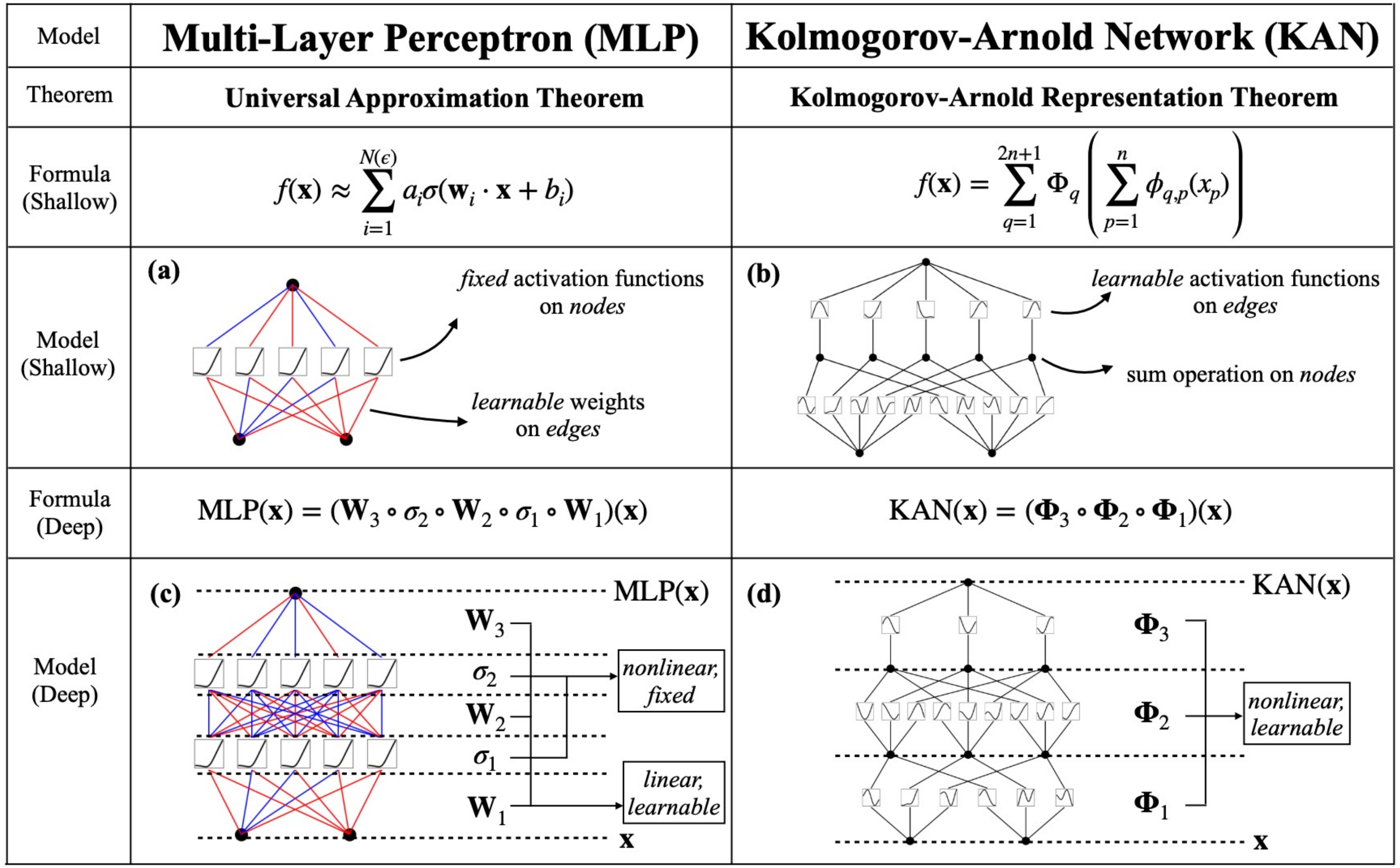
$$x_{l+1,j} = \sum_{i=1}^{n_l} \tilde{x}_{l,j,i} = \sum_{i=1}^{n_l} \phi_{l,j,i}(x_{l,i}), \qquad j = 1, \cdots, n_{l+1}.$$

Deep KANs



$x_{2,1}$
$\tilde{x}_{1,1,1}$ $\tilde{x}_{1,1,2}$ $\tilde{x}_{1,1,3}$ $\tilde{x}_{1,1,4}$ $\tilde{x}_{1,1,5}$
$\phi_{1,1,1}$ $\phi_{1,1,2}$ $\phi_{1,1,3}$ $\phi_{1,1,4}$ $\phi_{1,1,5}$
$x_{1,1}$ $x_{1,2}$ $x_{1,3}$ $x_{1,4}$ $x_{1,5}$
$\tilde{x}_{0,1,1}$ $\tilde{x}_{0,1,2}$ $\tilde{x}_{0,2,1}$ $\tilde{x}_{0,2,2}$ $\tilde{x}_{0,3,1}$ $\tilde{x}_{0,3,2}$ $\tilde{x}_{0,4,1}$ $\tilde{x}_{0,4,2}$ $\tilde{x}_{0,5,1}$ $\tilde{x}_{0,5,2}$
$\phi_{0,1,1}$ $\phi_{0,2,1}$ $\phi_{0,3,1}$ $\phi_{0,4,1}$ $\phi_{0,5,1}$ $\phi_{0,1,2}$ $\phi_{0,2,2}$ $\phi_{0,3,2}$ $\phi_{0,4,2}$ $\phi_{0,5,2}$
$x_{0,1}$ $x_{0,2}$

MLP vs KAN



Model
Multi-Layer Perceptron (MLP)
Kolmogorov-Arnold Network (KAN)
Theorem
Universal Approximation Theorem
Kolmogorov-Arnold Representation Theorem
Formula (Shallow)
$f(\mathbf{x}) \approx \sum_{i=1}^{N(\epsilon)} a_i \sigma(\mathbf{w}_i \cdot \mathbf{x} + b_i)$
$f(\mathbf{x}) = \sum_{q=1}^{2n+1} \Phi_q \left(\sum_{p=1}^{n} \phi_{q,p}(x_p) \right)$
Model (Shallow)
(a)
fixed activation functions on *nodes*
learnable weights on *edges*
(b)
learnable activation functions on *edges*
sum operation on *nodes*
Formula (Deep)
$\mathrm{MLP}(\mathbf{x}) = (\mathbf{W}_3 \circ \sigma_2 \circ \mathbf{W}_2 \circ \sigma_1 \circ \mathbf{W}_1)(\mathbf{x})$
$\mathrm{KAN}(\mathbf{x}) = (\mathbf{\Phi}_3 \circ \mathbf{\Phi}_2 \circ \mathbf{\Phi}_1)(\mathbf{x})$
Model (Deep)
(c)
MLP(x)
$\mathbf{W}_3$
σ_2
$\mathbf{W}_2$
σ_1
$\mathbf{W}_1$
x
nonlinear, fixed
linear, learnable
(d)
KAN(x)
$\mathbf{\Phi}_3$
$\mathbf{\Phi}_2$
$\mathbf{\Phi}_1$
x
nonlinear, learnable

Training KANs

Addition of a residual connection b(x):

$$\phi(x) = w_b b(x) + w_s \text{spline}(x).$$

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

Spline defined as before. c_is are trainable.

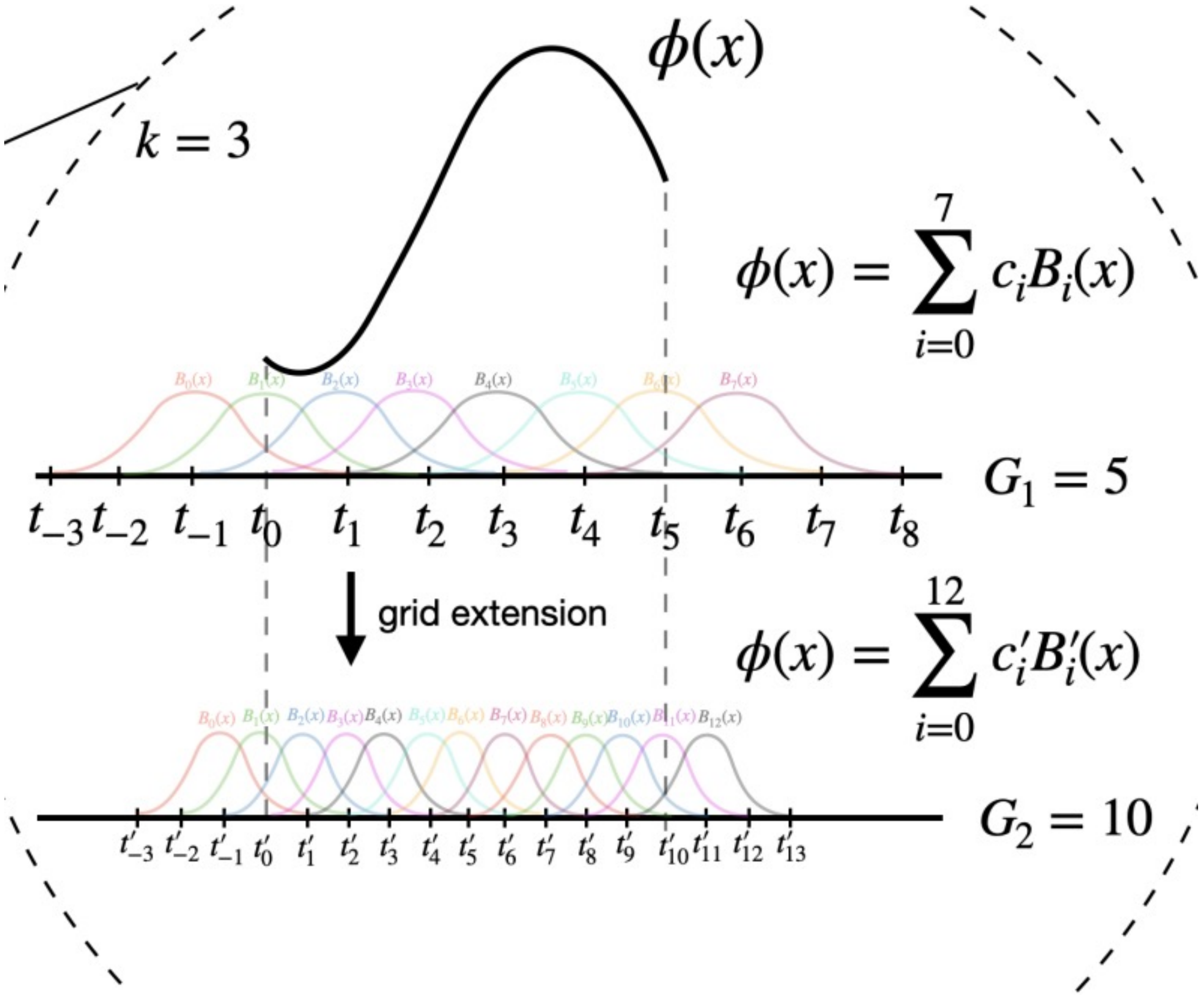
$$\text{spline}(x) = \sum_i c_i B_i(x)$$

Parameter count

- For a network of L layers, each of width N and a spline of order G intervals
  - KANs – $O(N^2LG)$
  - MLPs – $O(N^2L)$
- KANs usually require much smaller N than MLPs

Accuracy: Grid Extension

- Splines can be made arbitrarily accurate by increasing the grid size
- Optimization via least-squares algorithm



Accuracy: Grid Extension

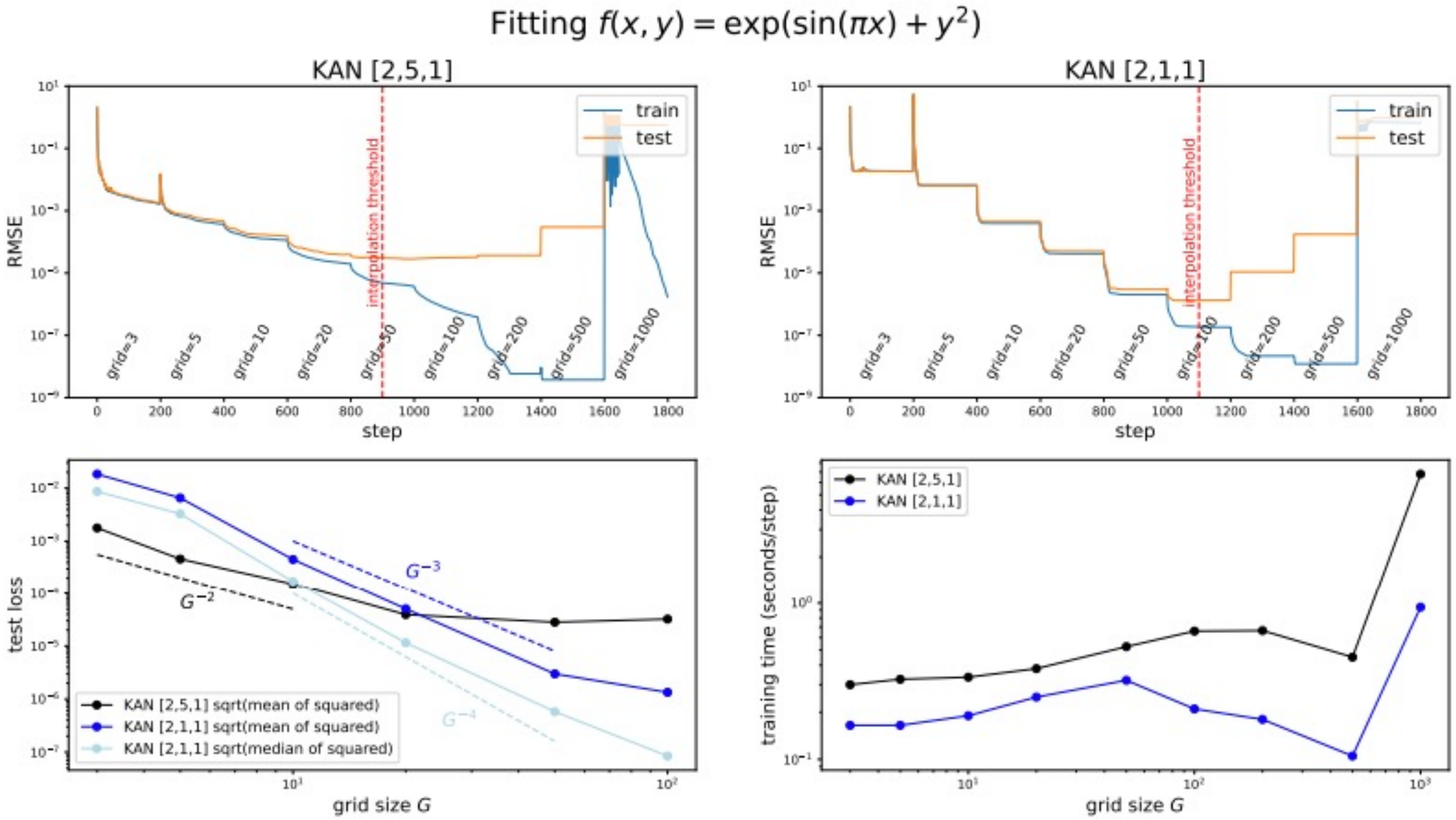


Figure 2.3: We can make KANs more accurate by grid extension (fine-graining spline grids). Top left (right): training dynamics of a $[2, 5, 1]$ ($[2, 1, 1]$) KAN. Both models display staircases in their loss curves, i.e., loss suddenly drops then plateaus after grid extension. Bottom left: test RMSE follows scaling laws against grid size G. Bottom right: training time scales favorably with grid size G.

Interpretability

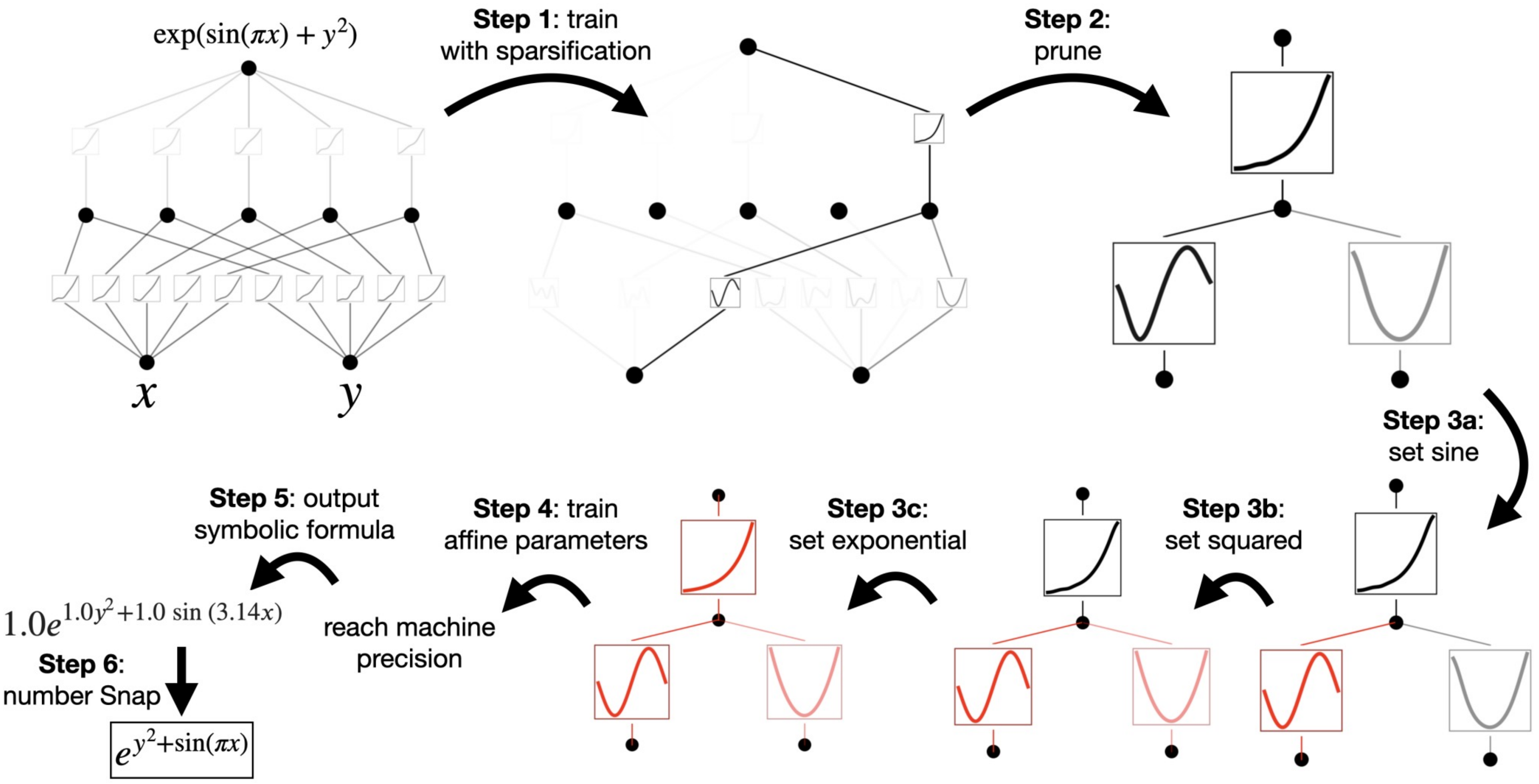


Figure 2.4: An example of how to do symbolic regression with KAN.

Continual Learning

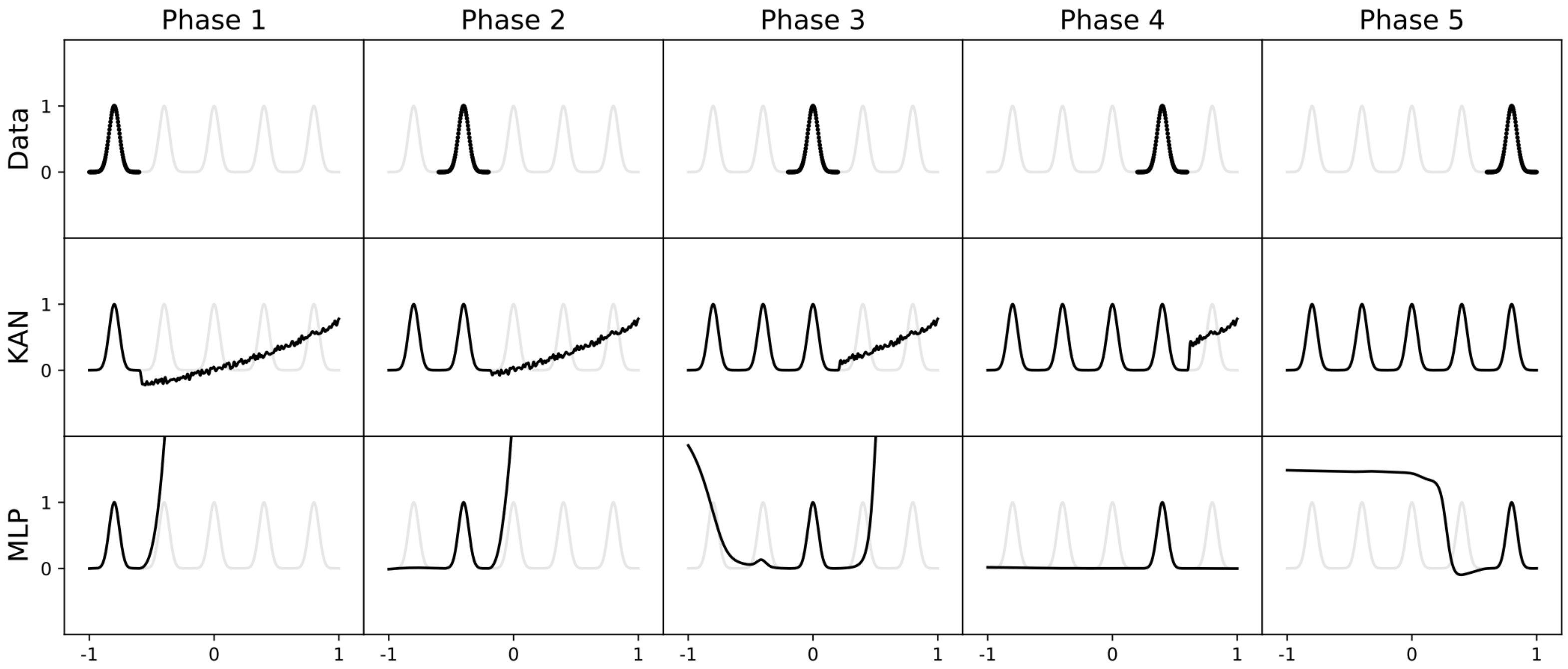


Figure 3.4: A toy continual learning problem. The dataset is a 1D regression task with 5 Gaussian peaks (top row). Data around each peak is presented sequentially (instead of all at once) to KANs and MLPs. KANs (middle row) can perfectly avoid catastrophic forgetting, while MLPs (bottom row) display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

Thank you!