KAN: Kolmogorov–Arnold Networks

Ziming Liu[1,4] Yixuan Wang[2] Sachin Vaidya[1] Fabian Ruehle[3,4] James Halverson[3,4] Marin Soljacic[1,4] Thomas Y. Hou[2] Max Tegmark[1,4]

[1]Massachusetts Institute of Technology
[2]California Institute of Technology
[3]Northeastern University
[4]The NSF Institute for Artificial Intelligence and Fundamental Interactions

CBMM Journal Club (14/05/2024)

Agenda

Overview

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 (*nonlinear, fixed*), $\mathbf{W}_2$, σ_1, $\mathbf{W}_1$ (*linear, learnable*); x	(d) KAN(x); $\mathbf{\Phi}_3$, $\mathbf{\Phi}_2$ (*nonlinear, learnable*), $\mathbf{\Phi}_1$; x

Overview

- Kolomogorov-Arnold Networks (KANs) are proposed as a promising altnerative to Multi-Layer Perceptrons (MLPs).
- Unlike MLPs, KANs have *learnable* activation functions on *edges*.
- KANs offer simpler and more interpretable structures than MLPs
- KANs offer more accuracy than MLPs for low-dimensional structures
- KANs are not trainable through batch computation, and therefore expensive

Kolomogorov-Arnold Representation Theorem

Theorem

If f is a multivariate continuous function, then f can be written as a finite composition of continuous unary functions and the binary operation of addition. More specifically, for a smooth $f : [0,1]^n \to \mathbb{R}$,

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

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

- Suppose we have a multivariate function $f(x_1, \cdots, x_n)$.
- The theorem says that we can express this function as the sum of a series of uniary functions, $\phi_{p,q}(x_p) : [0, 1] \to \mathbb{R}$.
- We can perfectly approximate f with $2n + 1$ uniary functions for each x_i.

Suppose we have an arbitrary dataset

a	b	y
1	2	3
2	4	6
3	6	9

- Instead of writing the multivariate form,

$$y = f(a, b, c),$$

- the theorem allows us to write,

$$y = \boldsymbol{f}(\phi_a(a) + \phi_b(b)).$$

- Or more explicitly,

$$y = \sum_{i=1}^{5} f_i(\phi_{a,i}(a) + \phi_{b,i}(b))$$

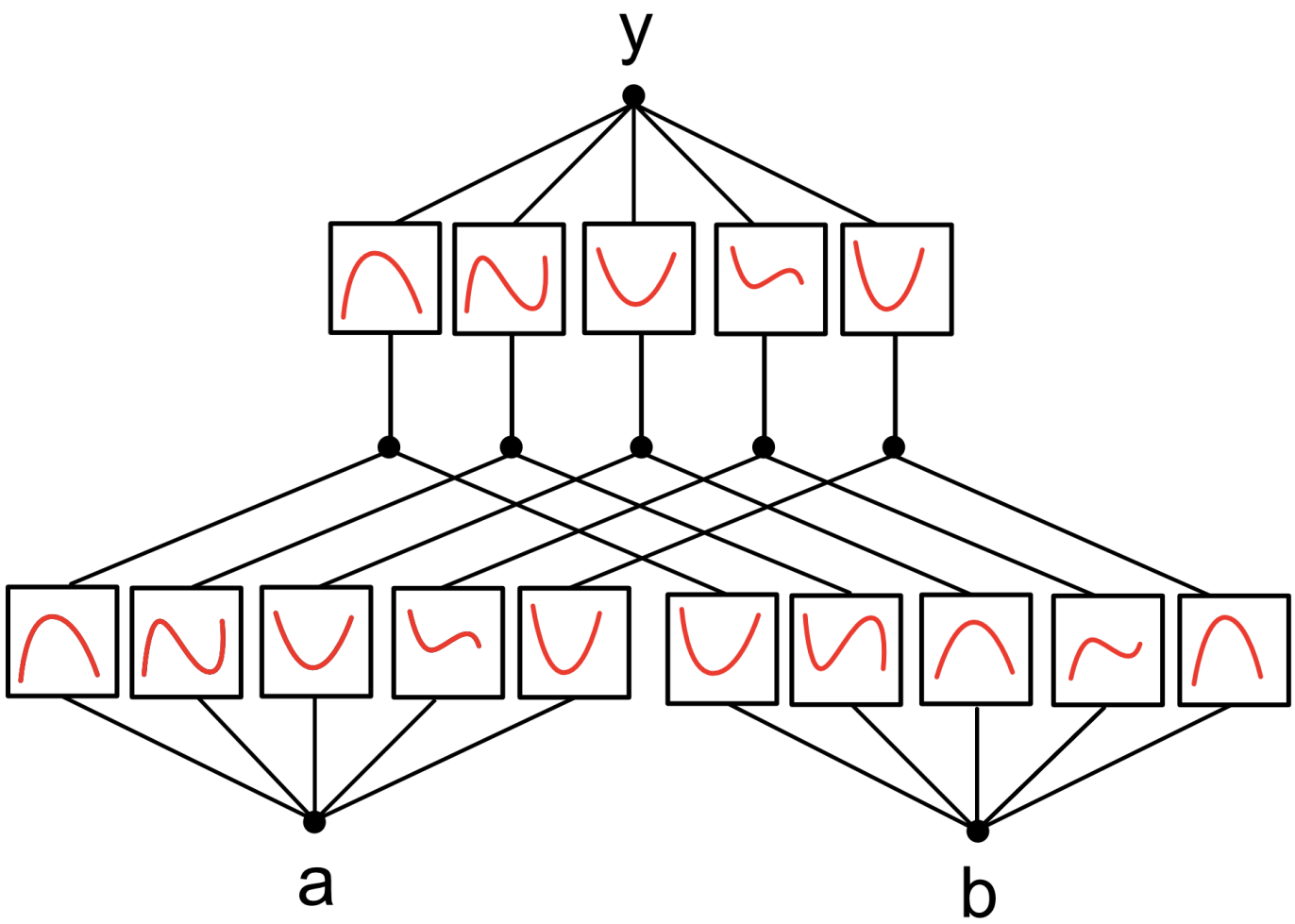

Figure: (2, 5, 1) KAN for a bivariate function.

- The original Kolomogorov-Arnold representation theorem in (1) only has two layers of non-linearities and $2n+1$ functions for each variable in f.
- **Problem**: this can result in functions that are non-smooth or fractal, which isn't suitable for back-propogation.
- **Solution**: modify the original theorem in (1) to allow representations to be arbitrarily wide and deep, allowing for smooth functions.
- We now have a new, modified version of the KAR Theorem in (1),

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

where m is large enough to allow $\boldsymbol{\Phi_q}$ and $\boldsymbol{\phi_{p,q}}$ to be smooth and differentiable.

- The action of one layer in the KAN has a matrix representation,

$$\underbrace{\begin{bmatrix} \phi_{1,1}(x_1) & \cdots & \phi_{1,n}(x_1) \\ \vdots & \ddots & \vdots \\ \phi_{m,1}(x_m) & \cdots & \phi_{m,n}(x_m) \end{bmatrix}}_{m \times n} \begin{bmatrix} 1 \\ \vdots \\ 1 \end{bmatrix} = \begin{bmatrix} \sum_{i=1}^{n} \phi_{1,i}(x_1) \\ \vdots \\ \sum_{i=1}^{n} \phi_{m,i}(x_m) \end{bmatrix}$$

  where ϕ are smooth activation functions parameterised as B-splines (stable, bounded, differentiable).
- The idea is to learn the parameters of the activiation functions $\phi_{i,j}$ through backpropogation.

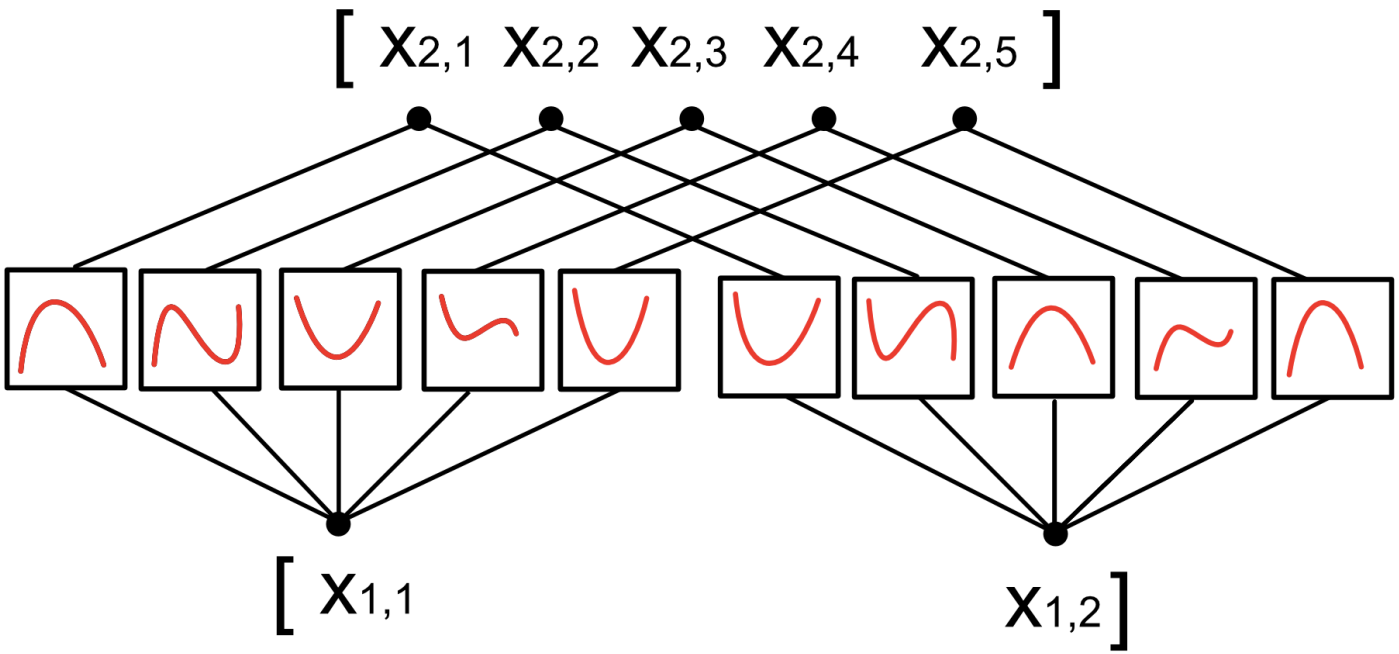

Figure: Example of a single KAN layer, recieving a 2-dimension input, and transforming it to a 5-dimension output.

Splines

- Suitable activation functions must be smooth and differentiable, but they must also be *expressive*.
- *Splines* are a collection of *curves* that are joined together to at *knots*.
- *Curves* are defined by *control points*, which we can parameterise through backpropogation
- By changing the position of control points, we can alter the shape of the curve.
- B-splines are a kind of spline that have very useful propertise here
  - C(2) continuous: first and second derivatives are continuous.
  - Local control: changing a control point only impacts a local area of the curve (prevents catastrophic forgetting)

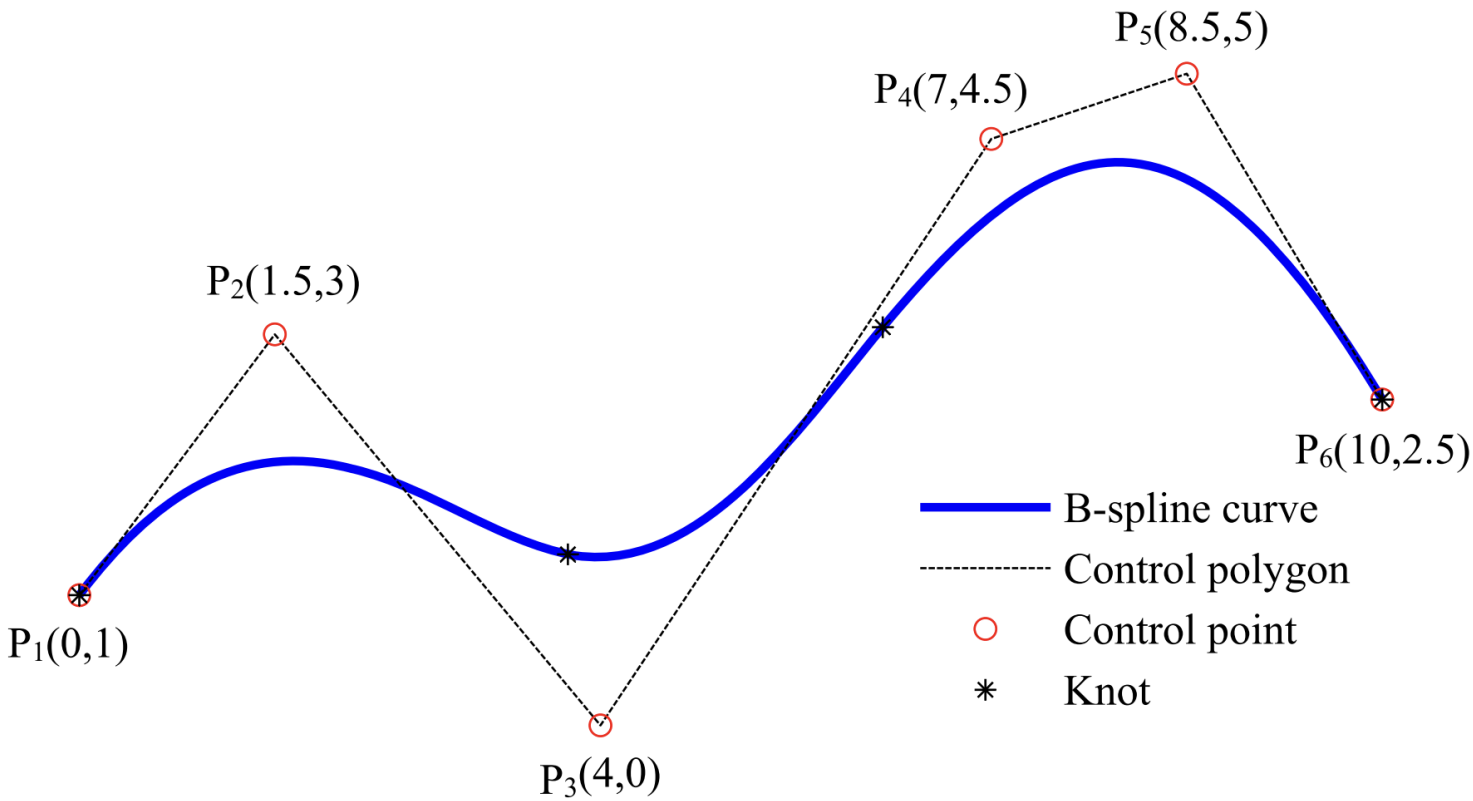

Figure: Example of a cubic B-spline.

Catestrophic forgetting

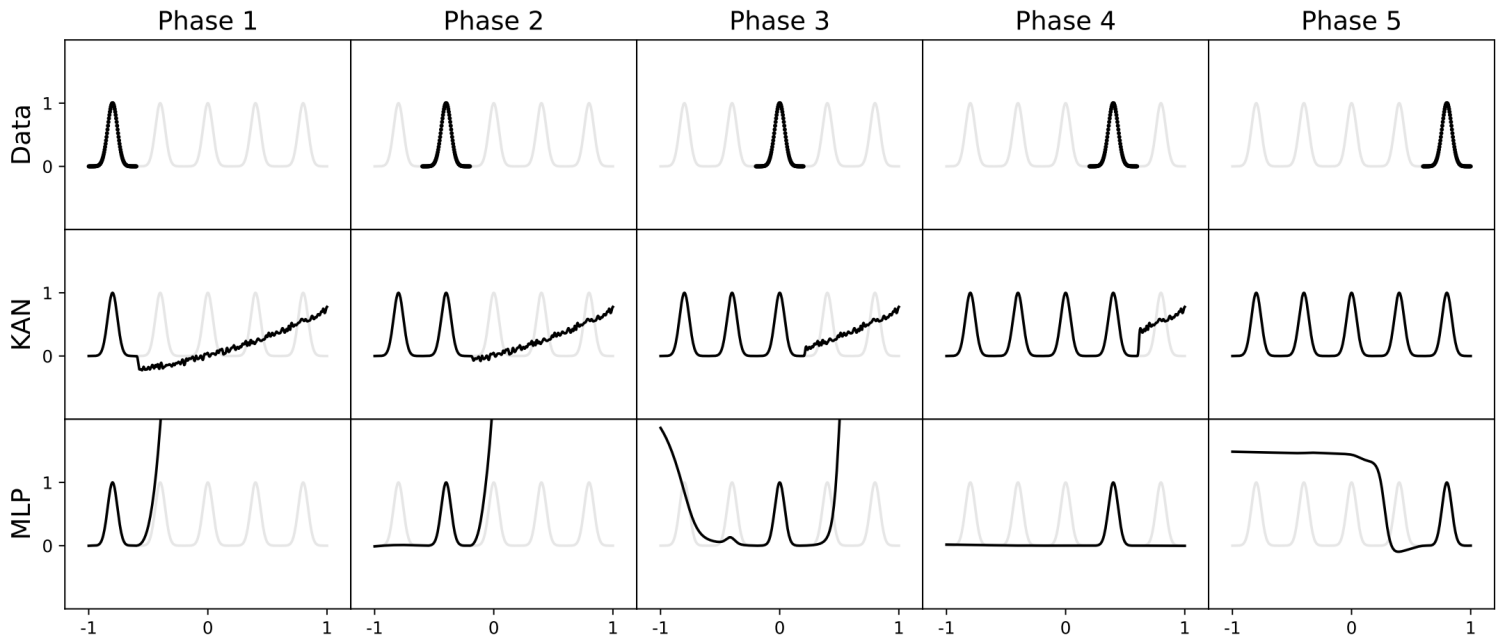

Figure: Illustration of 'catestrophic forgetting'.

Grid extension

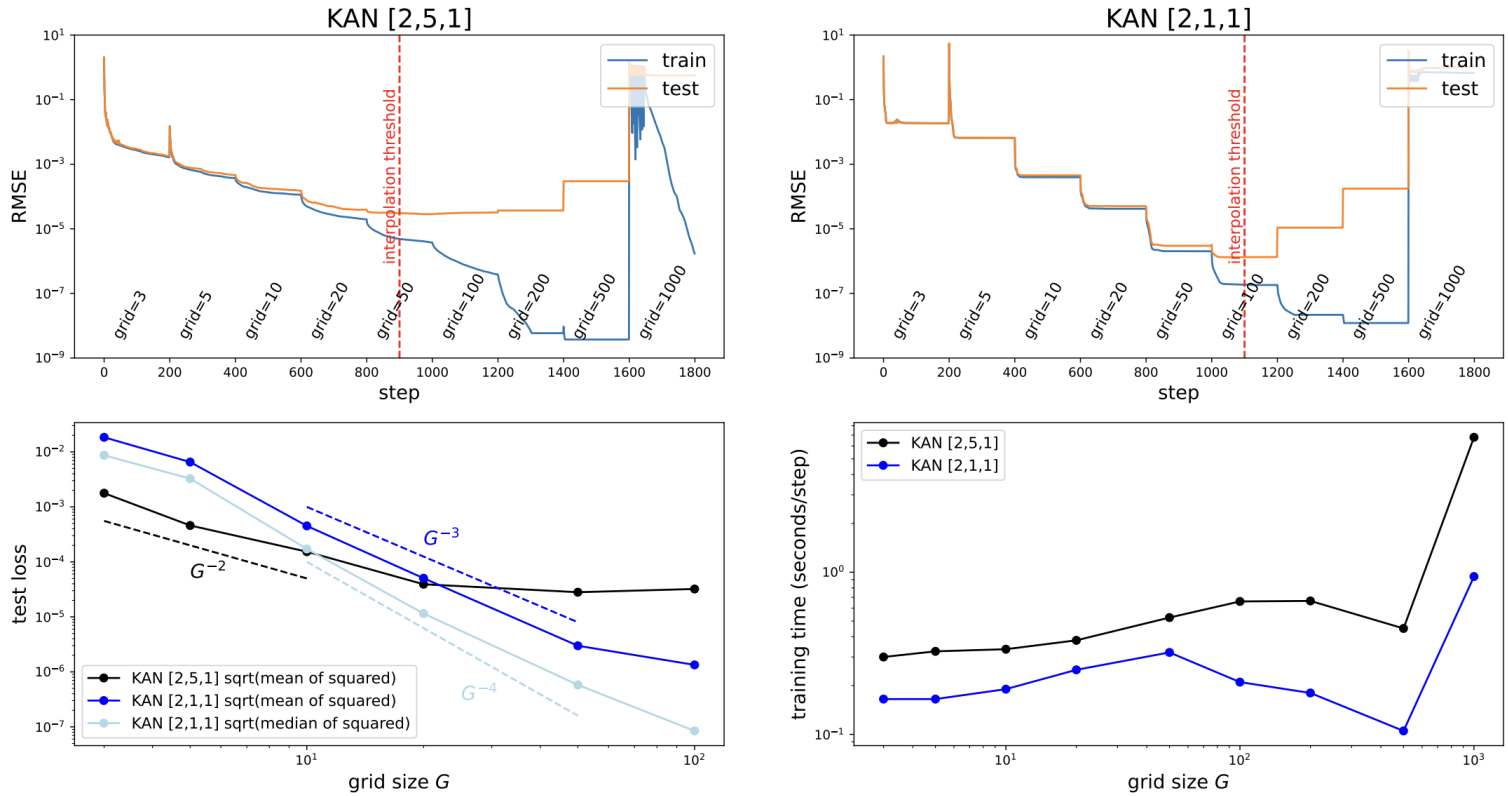

Figure: Illustration of *grid extension* in KANs, to fit $\exp(\sin(\pi x) + y^2)$. Grid extension improves accuracy through more fine-tuned spline grids.

Sparsification and interpretability

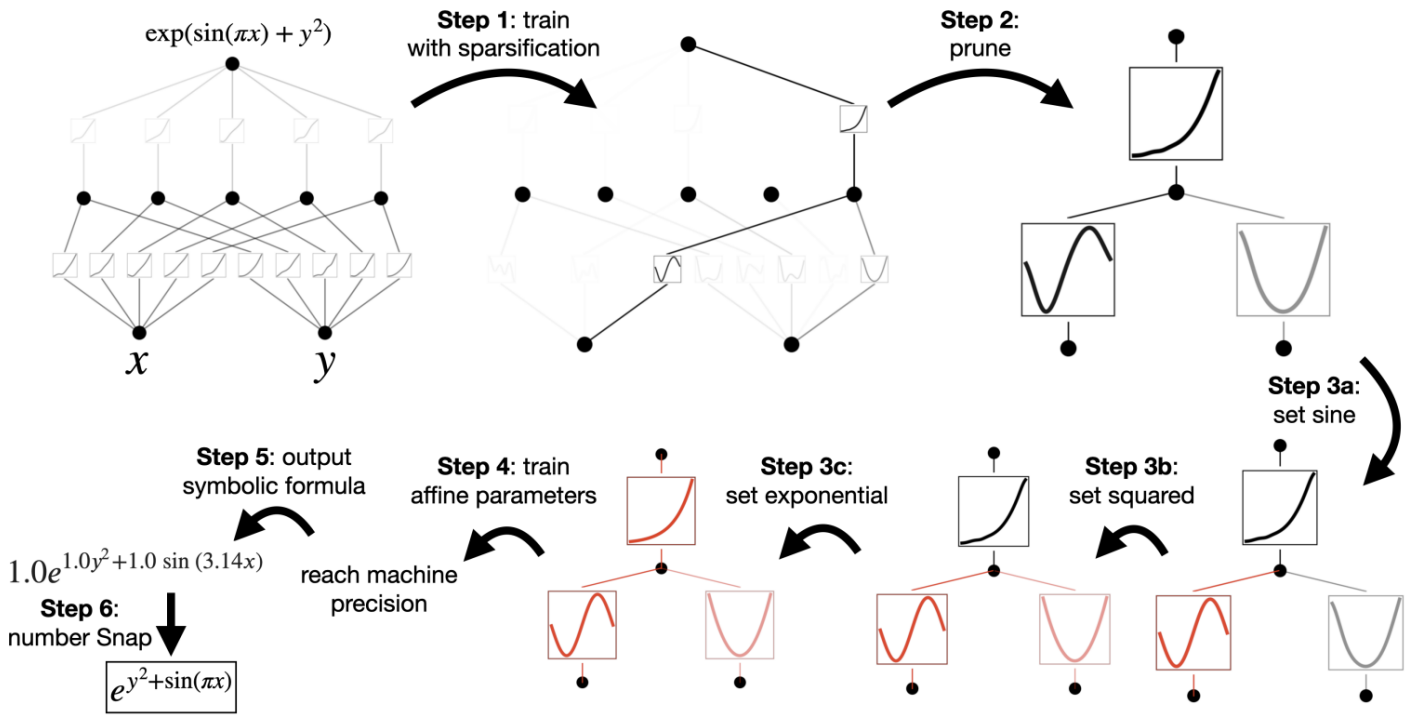

Figure: Illustration of *sparsification* in KANs, to fit $\exp(\sin(\pi x)+y^2)$.

Neural scaling of MSE

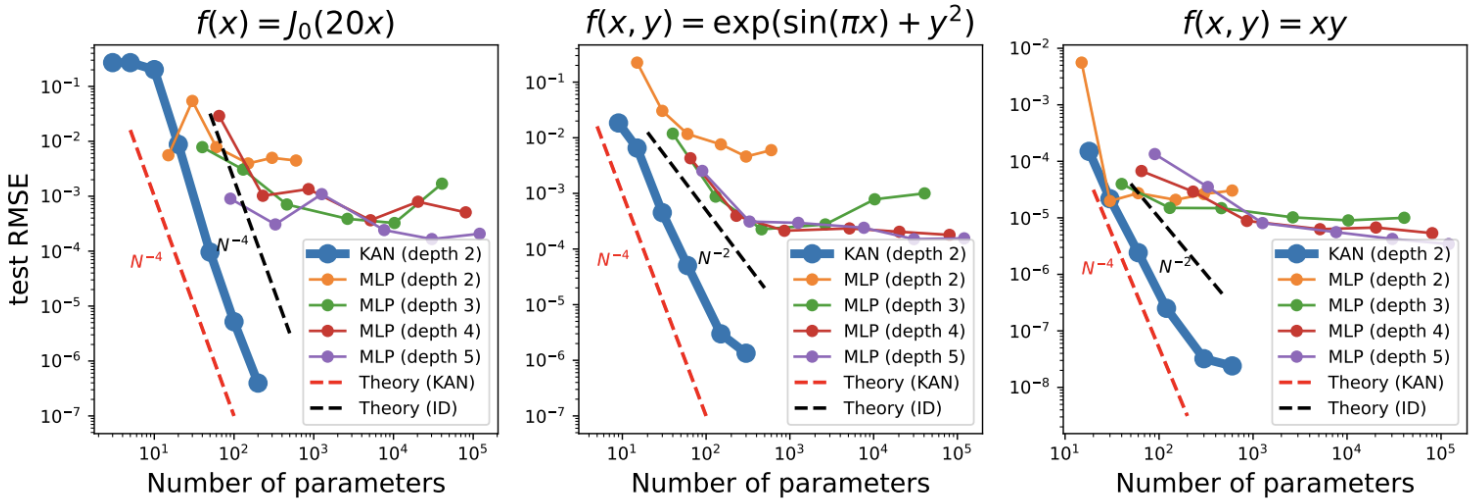

Figure: Reduction in test loss in KANs scales much better than in MLPs for toy datasets.

Discovering compositional structures

- KANs can discover compositional structures in input data.

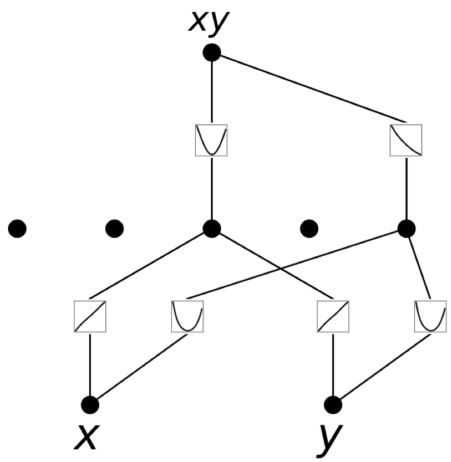

Figure: The KAN learns to calcualate xy by leveraging $2xy = (x+y)^2 - (x^2+y^2)$.

- KANs can discover compositional structures in input data.

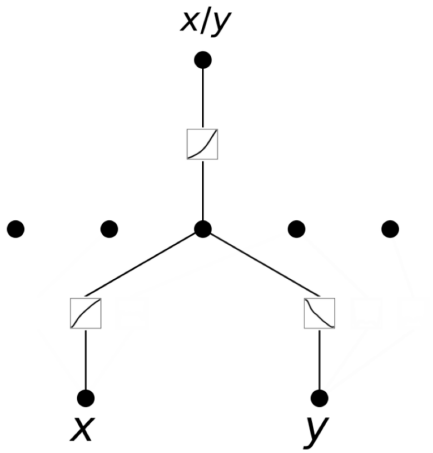

Figure: The KAN learns to calcualate x/y by leveraging $x/y = \exp(\log(x) - \log(y))$.

- KANs can discover compositional structures in input data.

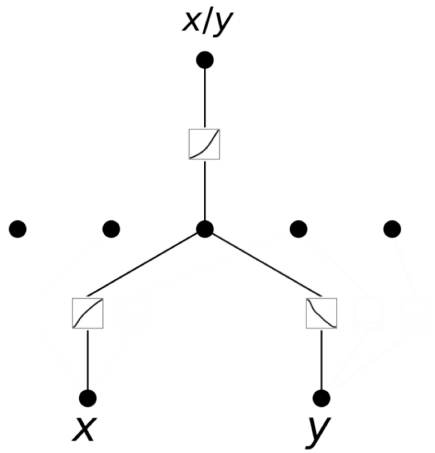

Figure: The KAN learns to calcualate x/y by leveraging $x/y = \exp(\log(x) - \log(y))$.

- KANs can discover compositional structures in input data.

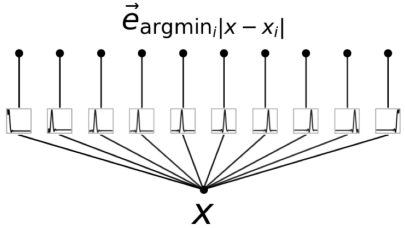

Figure: The KAN learns to convert a real number in $[0, 1]$ to its first decimal digit.

Unsupervised approaches

- KANs can discover dependence between variables in an unsupervised fashion.
- Idea: find a function $f(x_1, \cdots, x_n) = 0$, such that f is not the 0-function.
- If there is any dependence in $\{x_1, \cdots, x_n\}$, this will be evident in learned activation functions.
- Toy example, $x_1 = -2x_2$, x_3 is independent.

x_1	x_2	x_3
1	-2	5
2	-4	5
3	-6	5

- Can the KAN learn $f(\boldsymbol{x}) = 2x_1 + x_2 = 0$?

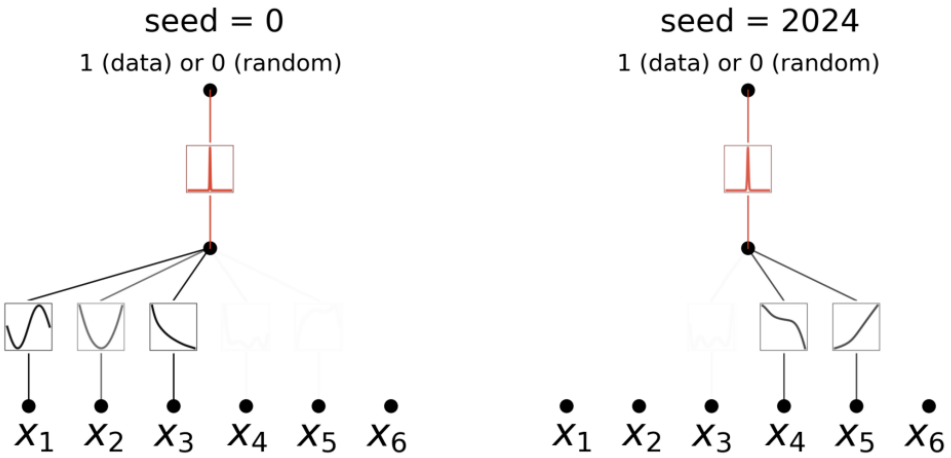

Figure: (x_1, x_2, x_3) are dependent variables such that $x_3 = exp(sin(x_1) + x_2^2)$; (x_4, x_5) are dependent variables with $x_5 = x_4^3$; x_6 is independent of the other variables. The KAN is able to learn both dependancies.

Summary

- Advantages:
  - Useful to discover symbolic/compositional structure
  - More interpretable than MLPs (for low dimension data)
  - More acurate than MLPs (for low dimension data)
  - Provides opportunity for continual learning
  - Reduces 'catastrophic forgetting'
  - Hypothesis testing/human interaction element
- Disadvantages:
  - Slow training (10x slower than MLP)
- Unclear:
  - How well do KANs perform at scale?

Should you use KANs or MLPs?