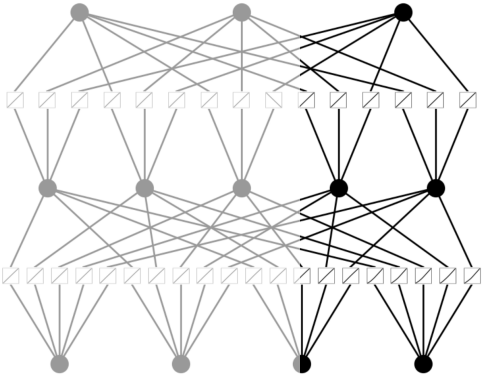
Kolmogorov Arnold Networks

Theoretical background, current approaches, potential next steps

Tatiana Boura
Stasinos Konstantopoulos



28 November 2024

Outline

- Part 1, Kolmogorov-Arnold Representation Theorem: Historical context; proof sketch; thoughts on the theorem's impact
- Part 2, Kolmogorov Arnold Networks: Definition; training algorithm
- Part 3, Why Kolmogorov Arnold Networks? Motivation; expected applications
- Part 4, Life after KAN: Extensions; DEG current work and plans

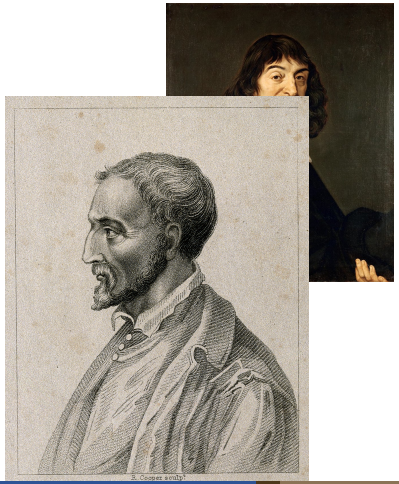
Part 1. **Kolmogorov-Arnold Representation Theorem**

The quadratic equation

- Methods for finding the positive roots of the positive-coefficients quadratic equation appear in Babylonian clay tablets from 21st c. BC.
- In the 9th c. al-Khwarizmi proves formulas that calculate the positive solution(s) as functions of positive, rational coefficients
- In the 13th c. Yang Hui solves quadratic equations with negative coefficients
- René Descartes' *La Géométrie* (1637) gives the quadratic formula in the form we know today

The cubic and quartic equations

- Ancient solutions or almost-solutions for specific cases: doubling the cube, Diophantine equations, conic sections
- 11th c. Omar Khayyam: geometric solution, realizes multiple solution exist, tries and fails algebraic formulation of the solution
- Scipione del Ferro solves $x^3 + mx = n$, not realizing it is general
- Girolamo Cardano realizes (1545) del Ferro's solution and Lodovico Ferrari's quartic solution are general
- Descartes gives the modern formulas (1637)



The end of the path

- Paolo Ruffini (1799, 1813) and Niels Henrik Abel (1824) prove that there is no algebraic solution to indeterminate equations of degree five or higher
- Galois Theory (1830, 1846) implies that there are equations of degree five and higher that do not have algebraic solution
  - Almost all polynomials cannot be solved
  - $\mathrm{x}^5 - \mathrm{x} - 1 = 0$ simplest unsolvable polynomial

Hilbert's 13th

- Since Tschirnhaus (1683) it is known that every 7th-degree can be algebraically reduced to $x^7 + ax^3 + bx^2 + cx + 1 = 0$
- Algebraic solutions are functions of three variables
- *Can this be expressed as the composition of a finite number of two-variable functions?* (Hilbert 1900, 1902)
- Following up on a series of post-Galois results on trying to understand high-order algebraic polynomial equations

Kolmogorov and Arnold's results

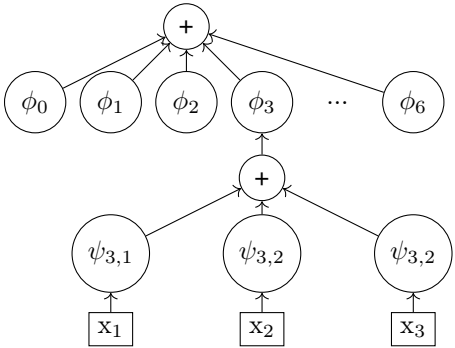
- Kolmogorov (1956): Every continuous multi-variable function can be expressed as the composition of a finite number of three-variable functions
- Arnold (1957): Only two-variable functions are really needed
- Kolmogorov (1957): Actually, the only two-variable function needed is addition
- This is a generalization of Hilbert's question, but the univariate functions are not necessarily algebraic

Kolmogorov-Arnold Representation Theorem

- Every multi-variate continuous function f can be written using only univariate functions and addition:

$$\mathrm{f}(\mathrm{x}_{\mathrm{i}=1..\mathrm{n}}) = \sum_{\mathrm{q}=0}^{2\mathrm{n}} \phi_{\mathrm{q}} \left(\sum_{\mathrm{p}=1}^{\mathrm{n}} \psi_{\mathrm{q,p}}(\mathrm{x}_{\mathrm{p}}) \right)$$

- In other words, addition is the only real multi-variate function, everything else is syntactic sugar

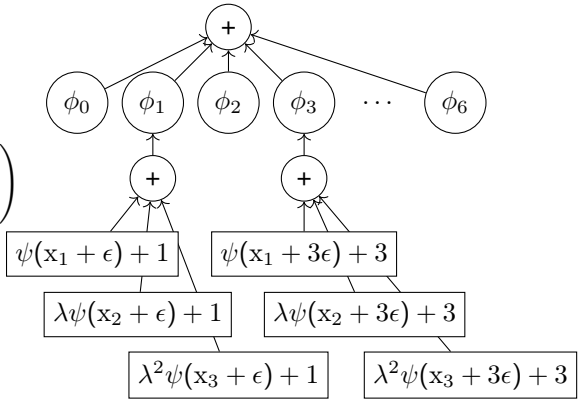


Sprecher Formulation

Sprecher (1972) gives an alternative formulation with a single function at the lower layer:

$$\mathrm{f}(\mathrm{x}_{\mathrm{i}=1..\mathrm{n}}) = \sum_{\mathrm{q}=0}^{2\mathrm{n}} \phi_{\mathrm{q}} \left(\sum_{\mathrm{p}=1}^{\mathrm{n}} \lambda^{\mathrm{p}-1} \psi(\mathrm{x}_{\mathrm{p}} + \mathrm{q} \cdot \epsilon) + \mathrm{q} \right)$$

where ϕ_{q} are continuous, ψ is monotonically increasing and Lipschitz-1 continuous, λ and ψ depend only on n and not on f, and ϵ is any non-zero constant

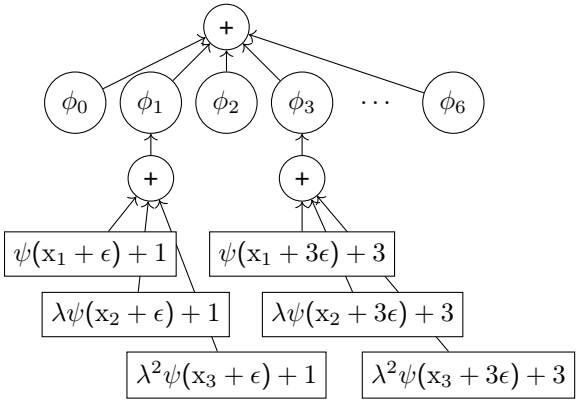


Some Thoughts and Intuitions

- The theorem premises that x_i are in the unit interval—this is needed for some of the constructions in the proof and is not a real restriction
- The theorem proves the existence of the ϕ and ψ functions, does not tell you how to construct them
- We know very little about ϕ (continuous) and slightly more (but still little) about ψ (monotonic increasing and Lip-1 continuous)
- It is unlikely that these functions can be defined by a neat-looking closed formula

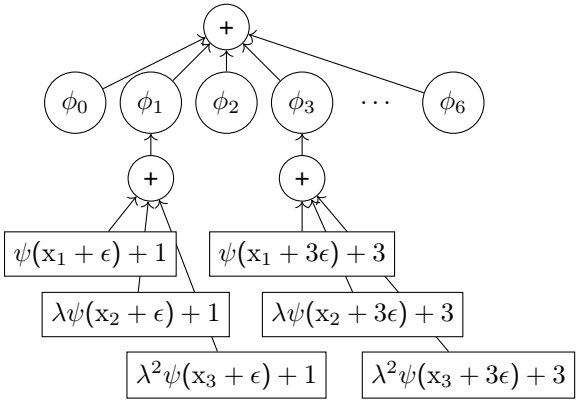
Some Thoughts and Intuitions

- Sprecher notes that he uses λ^{p} but other non-linear sequences also work
- Feels like the n values in the unit interval are 'packed' into a single value in $\mathcal{R}$ and then 'decoded' by the ϕ functions
  - Note how λ depends on n



Some Thoughts and Intuitions

- ψ has many 'intervals' and translating by $\mathrm{q} \cdot \epsilon$ lands $\mathrm{x_i}$ into the right interval
  - hacks the multiple ψ_i functions into one that behaves differently in each interval
- The lower level 'packs' the effect of f on $\mathrm{x_i}$ for different values of the other inputs $\mathrm{x_{j \neq i}}$
- The upper level decodes to deliver the output



Proof Sketch

The actual proof is between E^{n} and $\mathrm{E}^{\mathrm{n}-1}$, which gives inductively that a function in E^{n} can be represented with functions in E.
To allow for nice figures, we will show that for any $\mathrm{f} : \mathrm{E}^2 \to \mathcal{R}$ there are $\phi_{0..4} : \mathcal{R} \to \mathcal{R}$ and $\psi_{0..4,1..2} : \mathrm{E} \to \mathcal{R}$ such that:

$$\mathrm{f}(\mathrm{x}_1, \mathrm{x}_2) = \sum_{\mathrm{q}=0}^{4} \phi_{\mathrm{q}} \left(\psi_{\mathrm{q},1} \left(\mathrm{x}_1 \right) + \psi_{\mathrm{q},2} \left(\mathrm{x}_2 \right) \right)$$

The proof proceeds in three steps:

- Cover the unit square
- Construct the inner functions ψ
- Construct the outer functions ϕ

Proof Sketch Step 1: Cover the unit square

For any integer $\mathrm{k} > 0$, we can construct k non-overlapping intervals in E with length less than $1/\mathrm{k}$ As $\mathrm{k} > 0$ gets larger, the length of each interval gets smaller; at the limit the length of each interval will approach zero.

Fix a value for k.

Let $\mathrm{A_{k,1}}$ be the union of such intervals for $\mathrm{p} = 1$ (i.e., for the first input of f).

We repeat the process to make one such union $\mathrm{A^q_{k,1}}$ for each value of $\mathrm{q} = 0..2\mathrm{n}$. All $\mathrm{A^q_{k,1}}$ are one *family* and all $\mathrm{A^q_{k,2}}$ are another family. There are $\mathrm{n} = 2$ families.

Proof Sketch Step 1: Cover the unit square

Every $\mathrm{x} \in \mathrm{E}$ is contained in no less than $2\mathrm{n}$ (here, 4) members of the family

The families can always be constructed so that the gaps of their members do not intersect:

$\mathrm{A}^0_{\mathrm{k},1}$	x	x	x	x	x	x		x	x	x	x	x	x		x
$\mathrm{A}^1_{\mathrm{k},1}$		x	x	x	x	x	x		x	x	x	x	x	x	
$\mathrm{A}^2_{\mathrm{k},1}$	x		x	x	x	x	x	x		x	x	x	x	x	x
$\mathrm{A}^3_{\mathrm{k},1}$	x	x		x	x	x	x	x	x		x	x	x	x	x
$\mathrm{A}^4_{\mathrm{k},1}$	x	x	x		x	x	x	x	x	x		x	x	x	x

so that any point in E will be contained in at most one gap.

Proof Sketch Step 1: Cover the unit square

- Replicate the family $\mathrm{A}^{\mathrm{q}}_{\mathrm{k},1}$ into $\mathrm{A}^{\mathrm{q}}_{\mathrm{k},2}$
- For each q, constuct $\mathrm{S}^{\mathrm{q}}_{\mathrm{k}}$, the union of the non-overlapping squares $\mathrm{A}^{\mathrm{q}}_{\mathrm{k},1} \times \mathrm{A}^{\mathrm{q}}_{\mathrm{k},2}$
- Any point in E^2 will be contained in at most one gap in each dimention, so will be contained in at least $2\mathrm{n}+1-\mathrm{n}=\mathrm{n}+1$ intervals

The system of all rectangles $\mathrm{S}^{\mathrm{q}}_{\mathrm{k}}, \mathrm{q}=0..2\mathrm{n}$ covers the unit square E^2 so that every point in E^2 is covered at least $\mathrm{n}+1$ times.

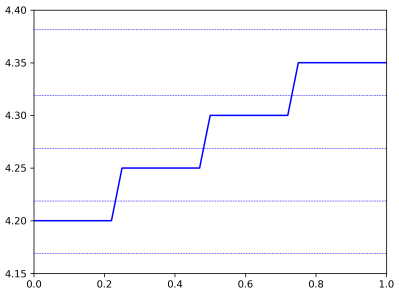
Proof Sketch Step 2: Construct the inner functions

For given k, q, it is always possible to construct a series of constants λ_i such that:

$$\lambda_\mathrm{i} < \lambda_{\mathrm{i}+1} \leq \lambda_\mathrm{i} + \frac{1}{2^\mathrm{k}}$$

and a function ψ_1 mapping inputs from the $\mathrm{A}^\mathrm{q}_{\mathrm{k},1}$ intervals to the corresponding λ intervals.

Any continuous function will do, but let's say its a step function with linear connections across gaps, so that it's continuous and monotonic increasing.



For $\mathrm{k} = 4$, four non-overlapping intervals and the corresponding 'lanes' defined by the λ series.

Proof Sketch Step 2: Construct the inner functions

Construct a similar series μ_{i} for the second dimension. Then:

$$\lambda_{\mathrm{i}} + \mu_{\mathrm{i}} < \lambda_{\mathrm{i}+1} + \mu_{\mathrm{i}+1} \leq \lambda_{\mathrm{i}} + \mu_{\mathrm{i}} + 2\frac{1}{2^{\mathrm{k}}}$$

so an $\epsilon_{\mathrm{k}} < 2^{-\mathrm{k}}$ can be found such that the closed intervals $[\lambda_{\mathrm{i}} + \mu_{\mathrm{i}}, \lambda_{\mathrm{i}} + \mu_{\mathrm{i}} + 2\epsilon_{\mathrm{k}}]$ do not intersect.

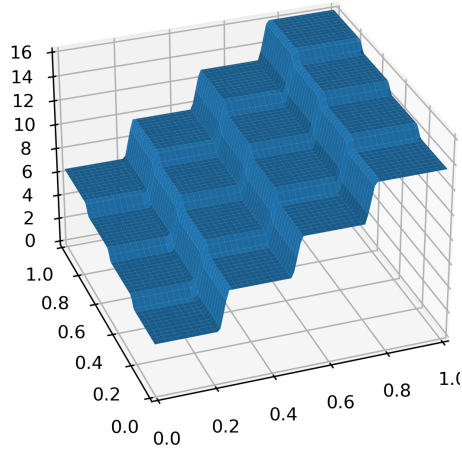
Since they do not intersect, we can apply the same logic as before to construct two functions ψ_1, ψ_2 such that their addition maps elements from the S_{k} squares we constructed before, to values within these intervals:

$$\text{For each } \mathrm{x}_1, \mathrm{x}_2 \in \mathrm{S}^{\mathrm{q}}_{\mathrm{k}}, \psi_1(\mathrm{x}_1) + \psi_2(\mathrm{x}_2) \in [\lambda_{\mathrm{i}} + \mu_{\mathrm{i}}, \lambda_{\mathrm{i}} + \mu_{\mathrm{i}} + 2\epsilon_{\mathrm{k}}]$$

Proof Sketch Step 2: Construct the inner functions

There are many solutions, but Sprecher's (1972) formulation gives us something we can visualize:

A grid of non-overlapping 'platforms', linearly connected at the gaps, all having different z, so that each z maps to exactly one S_k or to a gap.



Proof Sketch Step 2: Construct the inner functions

Reminder from Step 1:

- For each $\mathrm{q} = 0..2\mathrm{n}$, we constucted a family of non-overlapping squares $\mathrm{S}_{\mathrm{k}}^{\mathrm{q}}$
- Every point in E^2 is covered by at least $\mathrm{n}+1$ squares

So we repeat all the above for all q to create $2\mathrm{n}+1$ different n-tuples of ψ functions.

- For each q, we have created a mapping from some values of $\psi_1(\mathrm{x}_1) + \psi_2(\mathrm{x}_2)$ to a $\mathrm{S}_{\mathrm{k}}^{\mathrm{q}}$. The rest of the possible values of $\psi_1(\mathrm{x}_1) + \psi_2(\mathrm{x}_2)$ might be over a gap, but:
- For each $\mathrm{x}_1, \mathrm{x}_2 \in \mathrm{E}^2$, at least $\mathrm{n}+1$ of the q families are not gaps

Proof Sketch Step 3: Construct the outer functions

All the pieces are now in place:

- We have an 'index' from $\psi_1(\mathrm{x}_1) + \psi_2(\mathrm{x}_2)$ to squares restricting the possible values of $\mathrm{x}_1, \mathrm{x}_2$. The size of the squares gets smaller as k gets larger.
- We can construct a function f_k that simply 'memorizes' how to approximate the value of the original f from the sum of ϕ_p
  - all pairs in the unit square are covered
  - f_k knows the neighbourhood of the $\mathrm{x}_1, \mathrm{x}_2$ values despite the fact that only sees sums
  - f_k can estimate f with some error bounded by an expression denominated by k
- At the limit, there is no error

Unlike the ψ functions, we have no intuitions whatsoever about what the ϕ functions look like or how to construct them.

Some More Thoughts

There are many ways to pack two real numbers into one and then define the function that unpacks the original numbers and applies f. Here is one packing schema:

$$\begin{array}{l} 0.42 \\ 0.31415926\ldots \\ \hline 0.4321040105090206\ldots \end{array}$$

The Kolmogorov-Arnold therom tells us that it is possible to find a **continuous** function that does the same

- 'Continuous' sounds like a positive quality, but many 'pathological' functions are nominally continuous
- Under this light, we consider the claim that addition is the only multi-variate function strictly speaking valid, but over-hyped

Some More Thoughts

Kudos to Kolmogorov and Arnold for:

- Anticipating modern ML: As long as it fits the data, it doesn't matter if the model captures the actual structures that generate the data
- Anticipating modern ML II: In the executive summary, over-hype the significance of the result
- Anticipating that multiple arguments can be passed as one complex record a couple of years before COBOL
- Making people reconsider the definition of function complexity

Post-KART

- Hilbert's intuition about not being able to represent more complex functions by algebraically combining simpler functions remains valid:
  - It obviously remains open for algebraic functions
  - Its essense holds for any function, except that the K-A Representation Theorem proves that using the number of arguments to define complexity is wrong
  - Vitushkin (2004) gives a compelling argument
- Hecht-Nielsen (1987) notes the similarity between Sprecher's formulation and NNs and speculates that NN training can be used to construct ϕ, but Girosi & Poggio (1989) point out that we know ϕ are not smooth (Vitushkin & Henkin, 1967) and cannot be learnt.
- Schmidt-Hieber (2021) gives a good overview of the ways the KART has interacted with the ML community.

Part 2. **Kolmogorov-Arnold Networks**

Neural Networks are Function Approximators

- MLPs are inspired by the Universal Approximation Theorem
  - Leshno et al. (1993): *A standard multilayer FNN can approximate any continuous function to any degree of accuracy iff the network's activation functions are not polynomial*
- Can we be inspired by KART?

Neural Networks are Function Approximators

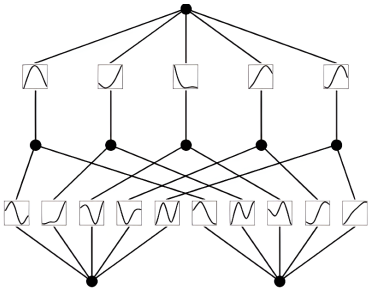
- Can we be inspired by KART?
  - KART claims that learning a high-dimensional function reduces to learning a polynomial number of 1D functions then, why **inspired**? . . .
  - . . . these 1D functions can be non-smooth (aka *not learnable!*)
  - . . . using only 2-layer nonlinearities and $2\mathrm{n} + 1$ terms in the hidden layer is pretty restrictive
- *But*,
  - We can argue that most ML-concerning functions are smooth, and
  - We can just stack ϕ layers and pray
- So, we are no longer using KA(R)T but moving towards a KA(A)T
- This is how Kolmogorov-Arnold Networks (KANs) are born (Ziming Liu et al., 2024)

Architecture: High Level

- *Inspiration:* KART

$$\mathrm{f}(\mathrm{x}_1, \ldots, \mathrm{x}_\mathrm{n}) = \sum_{\mathrm{q}=0}^{2\mathrm{n}} \phi_\mathrm{q} \left(\sum_{\mathrm{p}=1}^{\mathrm{n}} \psi_{\mathrm{q},\mathrm{p}}(\mathrm{x}_\mathrm{p}) \right)$$

- We just need to learn the univariate functions ψ and ϕ and perform addition
- ($\rightarrow$) A shallow KAN representing the above equation, with $\mathrm{n} = 2$ input variables x_p, $2\mathrm{n} + 1 = 5$ ϕ_s and $\mathrm{f}(\mathrm{x}_1, \mathrm{x}_2)$ as the output
- Learnable activation functions are positioned on edges and summation is performed on nodes



Architecture: Comparison with MLP

Shallow Formula

- Though KART ensures exact representation and UAT guarantees approximation, relaxing KART's assumptions positions both as function approximators
- UAT lacks guarantees on shallow MLP size, and KART provides no method to compute the relative functions
- MLP uses fixed activations on nodes and learnable weights on edges, while KAN employs fixed learnable functions on edges and summation on nodes

Architecture: Comparison with MLP

Deep Formula

- Shifting to deep formula to avoid UAT's and KART's restrictions
- KAN shifts non-linearity computation to the composition of ϕ_s rather than the weights and activation functions in MLPs
- KAN defines a *KAN layer* with n_{in}-D inputs and n_{out}-D outputs as a matrix of 1D functions, stacking these layers (note how we are drifting further away from KART)
- So, the dimensions of the KAN are *not* restricted to $[n, 2n + 1, 1]$

Architecture: Lower Level

- The learnable univariate functions are parametrized as a **B-spline (basis spline) curves** with learnable coefficients of local B-splines
- Splines are a transformation that maps control points to a smooth curve, aiming to generally follow the shape of these points
- Let us present the B-splines in depth

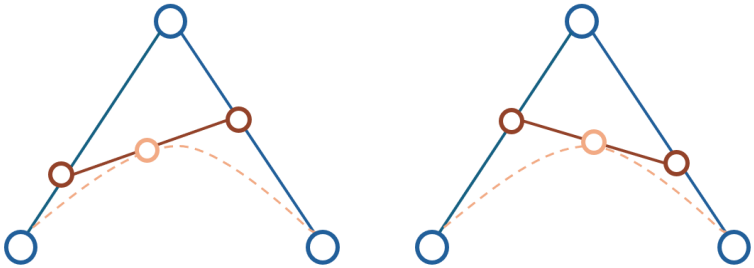
Bézier curves: Linear

To create a path between 2 points P_0 and P_1, we perform **linear interpolation** :

$$\mathrm{P}(\mathrm{t}) = (1 - \mathrm{t})\mathrm{P}_0 + \mathrm{t}\mathrm{P}_1$$

Bézier curves: Quadratic

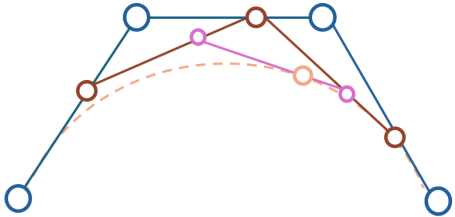
- If we add a third point P_2, then we have two lines connecting them and we can perform linear interpolation to them as well



- This curve is called a **quadratic** Bézier curve

Bézier curves: Cubic and higher

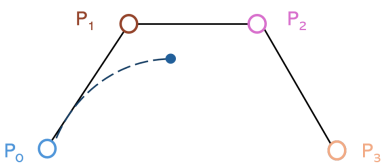
- In the same spirit, we can define the **cubic Bézier curve** (and any other degree, respectively)

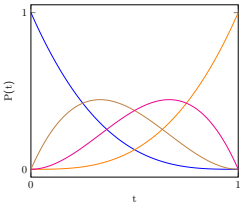


- We call the points **control points**, since the control the curvature

Bézier curves: Algorithms

- This repeated linear interpolation is called the **de Casteljau** algorithm
- A less expensive method for computing Bézier curves involves solving the **Bernstein polynomials** which are the linear interpolation equations expressed wrt the points



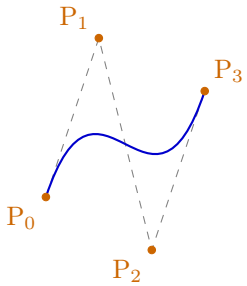


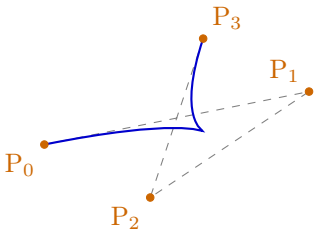
$$\begin{aligned} P(t) = \; & P_0 \cdot (-t^3 + 3t^2 - 3t + 1) + \\ & P_1 \cdot (3t^3 - 6t^2 + 3t) + \\ & P_2 \cdot (-3t^3 + 3t^2) + \\ & P_3 \cdot (t^3) \end{aligned}$$

- If we plot the factors of these polynomials we get a visualization of the influences of the control points (cf. right Illustration)

Bézier curves: (lack of) local control

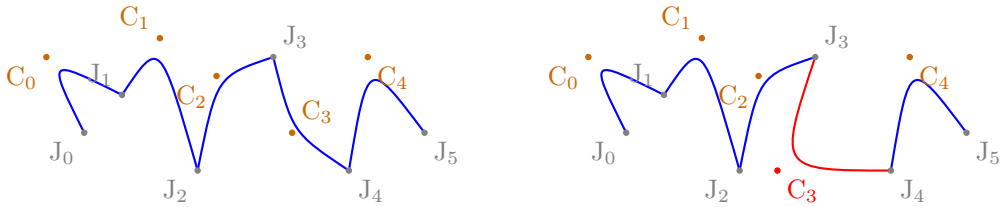
- One main downside of Bézier curves is the lack of local control
- This means that by changing **only one** control point, the curve can change drastically
- Pay attention because we will need this later on!





Bézier splines

- How to obtain local control? **Concatenate Bézier curves** → Bézier splines



- Additionally, this alleviates the issue of a high polynomial degree

Bézier splines: Continuity

- In the previous example the curve is not very smooth
- To fix that, we introduce the notion of **continuity**
- Bézier splines can have different levels of continuity at the joint
  - C^0-continuity refers to adjacent Bézier curves ending and starting at the same point
  - C^1-continuity requires C^0-continuity and matching first derivative vectors at the joint
  - C^2-continuity requires C^1-continuity and matching second derivative vectors at the joint
- For **smooth curves**, C^2-continuity is required

B-splines

- B-splines (basis splines) are C^2-continuous Bézier splines
- B-splines of order k are polynomial basis functions for splines of the same order, where, any spline can be represented as a linear combination of B-splines
- Additionally, there exists a unique combination of B-splines for any given spline (de Boor, 1978)
- B-splines are fitting candidates for performing the so-called **spline interpolation**

B-splines: Spline interpolation

- Spline interpolation is used to approximate functions by fitting splines, rather than using a single high-degree polynomial curve
- This process arises naturally from the formal definition of B-splines: it recursively defines each basis from a knot vector (Kaihuai, 1998), which indicates the positions where the splines should be connected
- Given grid G we define a spline S of k^{th} order as:

$$\mathrm{S}_{\mathrm{k,G}}(\mathrm{x}) = \sum_{\mathrm{i}} \mathrm{c_i B_{i,k}}(\mathrm{x}), \text{ where}$$

  $\mathrm{c_i}$ are the coefficients and $\mathrm{B_{i,k}}(\mathrm{x})$ are the corresponding basis functions
  - Liu et al. call grid what is usually called a uniform knot vector and coefficients what are usually called control points
- This process maximizes pre-computed elements (*minimizing possible learnable parameters*)

B-splines and KANs

- In KANs, each activation function is a k^{th} order B-spline, with a G-valued and uniformly spaced grid
- For each activation function what is left to learn is the coefficients of the spline
- Note that KAN also extends the grid on the edges of the spline, so that each grid point is influenced from the same number of polynomials
- Thus, for a KAN of width W and depth D the #training parameters is $\mathbb{O}(\mathrm{W}^2\mathrm{DG})$
- The respective number of parameters for an MLP is $\mathbb{O}(\mathrm{W}^2\mathrm{D})$, but KAN promises convergence with smaller widths

KANs: Implementation details

The authors used a few (questionable) tricks to train the network for optimal performance:

1. *Residual activation functions*:

$$\psi(\mathrm{x}) = \mathrm{w_b} \cdot \mathrm{silu}(\mathrm{x}) + \mathrm{w_s} \cdot \mathrm{spline}(\mathrm{x}), \text{ where } \mathrm{spline}(\mathrm{x}) = \sum_{\mathrm{i}} \mathrm{c_i B_i}(\mathrm{x})$$

(One can argue that this introduction of weights does not align with the initial claim)

2. *Initialization scales*: The network's training is sensitive to initialization, so an optimal setup is introduced

Part 3. **Why Kolmogorov-Arnold Networks?**

Grid extension

- For learning a B-spline, more basis functions result in a finer-grained curve
- A very useful feature of KANs is their ability to at first learn a coarse curve for a specific number of grid points and then, at any point, resume the training with more grid points
- The network does not need to be re-trained, but instead they employ a smart idea:

The coefficients of the fine-grained curve can be initialized from the coefficients of the coarse by minimizing the distance between the respective splines (over some distribution of input values)

Fixed activation functions

- KANs also give the option instead of using learnable activation functions, to use **fixed symbolic functions** (polynomials, exponentials etc.)
- This is useful for problems where we have an intuition about their internal functions
- However, we cannot simply set the activation function to be the exact symbolic formula, since its inputs and outputs may have shifts and scalings
- So, in this case, the network learns the parameters of the affine transformation

$$\mathrm{c} \cdot \psi_{\mathrm{fixed}}(\mathrm{a} \cdot \mathrm{x}_{\mathrm{in}} + \mathrm{b}) + \mathrm{d}$$

Interpretability: Motivation

- The main selling point of KANs is their interpretable structure, allowing **retrieval and examination of the learned activation functions**
- But, this is only useful when we a priori know the structure of the KAN and know what we expect to see
- For unknown problems, we stack KAN layers and experiment to find what works
- In this case, only some activation should be meaningful
- Then, how is the visualization considered interpretability?
- The idea is to start from a large enough KAN and train it with sparsity regularization followed by pruning

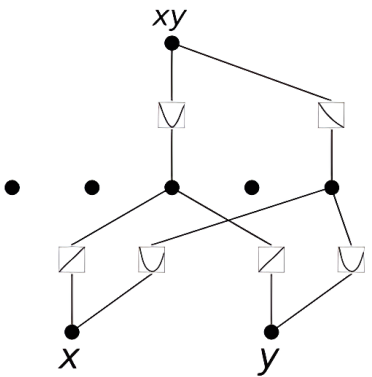
Interpretability: Sparsification

- In MLPs L1 regularization adds the absolute values of the model's weights to the loss function, promoting sparsity by encouraging smaller or zero weights
- Since KANs do not have linear weights, the L1-norm should be defined over the activation functions
- The L1 norm of an activation function (ψ) is its average magnitude over its inputs
- The L1 norm of a KAN layer (ϕ) is the sum of L1 norms of all ψ's
- Actually, L1-norm alone is insufficient, since the goal is to remove less informative or redundant features, not just to shrink weights, so an entropy penalty is introduced (layer-wise)
- So, the total training objective is the prediction loss plus L1 and entropy regularization of all KAN layers

Interpretability: Pruning

- After training with sparsification penalty, we can prune the network to a smaller subnetwork
- The incentive is to throw away the unimportant activation functions
- For each node, its incoming score is defined as the maximum L1 norm of the outputs of the prior KAN layer
- Its outgoing score is defined as the maximum L1 norm of the outputs of the current KAN layer
- A node is considered to be important if **both incoming and outgoing scores are greater than a threshold hyperparameter**

Interpretability: Example



- Function to be learned $\mathrm{f}(\mathrm{x},\mathrm{y}) = \mathrm{xy}$
- KAN computes:

$$(\mathrm{x}+\mathrm{y})^2 - (\mathrm{x}^2+\mathrm{y}^2)$$

- Note that:

$$(\mathrm{x}+\mathrm{y})^2 - (\mathrm{x}^2+\mathrm{y}^2) = 2\mathrm{xy}$$

Beating the COD (?)

- The **curse of dimensionality (COD)** refers to the exponential growth in volume associated with adding more dimensions to a space
- UAT tells us that theoretically, a neural network with one hidden layer can approximate any function, but as the dimensionality of the input increases, the number of neurons required grows exponentially
- This leads to a MLP that is very large and potentially computationally intractable
- On the other hand, KAN's approximation ability is more dependent on the number of control points in each activation function and less on the data's dimension
- So, KAN's size will not increase exponentially as the data dimensions increase, but its grid should become more fine-grained
- However, we should note that the current implementation of KANs is not as optimized as the one for MLPs, so in practice the training times for high-dimensional data are comparable

Continual Learning

- Catastrophic forgetting is a problem in MLPs
- When a MLPs is trained on one task and then shifted to being trained on another, the network will soon forget about how to perform the first task
- Do you recall the motivation behind using splines to approximate curves instead of using polynomials? Locality!
- Since spline bases are local, a sample will only affect a few nearby spline coefficients, leaving far-away coefficients intact
- This is desirable assuming faraway regions may have already stored information that we want to preserve
- By contrast, since MLPs usually use global activations any local change may propagate to regions far away

Part 4. **Life after KAN**

Extensions

Name	TLTR	Reference (2024)
KAN time series forecasting network	Vanilla KAN with shape [168,40,40,40,24] predicts 24 time-steps from 168	Vaca-Rubio et al.
Kolmogorov-Arnold Network Ordinary Differential Equations (KAN-ODEs)	Vanilla KAN used as an approximator for a dynamical system in the neural ODE framework	Koenig et al.
Wavelet Kolmogorov-Arnold Networks (Wav-KAN)	KAN with shape [28x28,32,10] using wavelets instead of B-splines; Inputs are complete MNIST images	Bozorgasl and Chen
Temporal Kolmogorov-Arnold Networks (TKAN)	Changes the output of the LSTM cell to $\sigma(\text{KAN}(\mathrm{x}))$	Genet and Inzirillo

Related Projects

MANOLO: Explainability, computational efficiency

- Explore KANs as explainable models
- Explore ways to reduce inefficiency

NavGreen: Maximizing efficiency of solar power usage

- Heat output, timeseries prediction
- Modelling a physical phenomenon, KAN might outperform Transformer

Christoforos' MSc thesis:

- Using prior domain knowledge as KAN activation functions

Plans

- Deeper integration of KAN and recursive time-series processing
- Mix prior and learned activation functions
  - Exploit prior knowledge where available
  - Give network flexibility to select & parameterize, but also to model residuals from scratch
- CL4-2025-03-DIGITAL-EMERGING-07:
  - Formal reasoning, mathematical problem-solving, confidence level estimation, long-term planning, and seamless adaptation to dynamic environments
  - Continual learning, seamlessly adapt, learn from changing conditions, and retain knowledge essential for operation in dynamic, real-world environments
  - Explainable AI and Trustworthy Decision-Making: causal inference, transparency, accountability

Thank you! Questions?

DATA
ENGINEERING
GROUP

Code, bibliography, slides:
`https://github.com/data-eng/kan`