UAB

Universitat Autònoma de Barcelona

Trabajo de Fin de Grado

Grado en Matemáticas

Del Teorema de Representación de Kolmogorov-Arnold a las Kolmogorov-Arnold Networks

David Estaún Dubra

Supervisor

Roberto Rubio Núñez

Año

2026

Convocatoria

Junio

Agradecimientos

Quiero expresar mi más sincero agradecimiento a mi tutor por toda la ayuda prestada durante la realización de este trabajo. Le agradezco especialmente su constante apoyo, su disponibilidad y la guía proporcionada a lo largo de todo el proceso, que han sido fundamentales para el desarrollo de este trabajo.

Índice general

Introducción **1**

1. Teorema de Representación de Kolmogorov-Arnold **2**
1.1. Preliminares . . . 2
1.2. Demostración del Teorema . . . 4
1.2.1. Funciones de separación racional . . . 4
1.2.2. Teoremas de aproximación y estabilidad . . . 8
1.2.3. Teoremas sobre las aproximaciones . . . 11
1.2.4. Teorema de Representación de Kolmogorov-Arnold . . . 12

2. Limitaciones del Teorema de Kolmogorov-Arnold **14**
2.1. Sobre la optimalidad del $2n+1$ . . . 14
2.2. Restricciones sobre las funciones de las matrices . . . 15

3. Kolmogorov-Arnold Networks **17**
3.1. De la teoría a la práctica: Motivación de las KANs . . . 17
3.2. Arquitectura de una KAN . . . 17
3.3. Anatomía de las aristas . . . 18
3.4. Actualización de los atributos . . . 22
3.5. Flujo de entrenamiento . . . 24
3.6. Aproximadores universales . . . 25
3.7. Regularización . . . 26

4. Implementación **29**
4.1. Referencias anteriores . . . 29
4.2. Tecnologías y paradigma de diseño . . . 29
4.3. Tensorización de las evaluaciones . . . 30

5. Experimentación y resultados **34**
5.1. Objetivo de las pruebas . . . 34
5.2. Configuración general . . . 34
5.3. Experimentación . . . 35

6. Bibliografía **41**

A. Teorema de Representación de Kolmogorov-Arnold **43**

B. Limitaciones del Teorema **47**

C. Kolmogorov-Arnold Networks **48**

D. Resultados **50**

Introducción

En el año 1900, David Hilbert formuló su célebre decimotercer problema, en el que cuestionaba la posibilidad de descomponer funciones continuas de varias variables en superposiciones de funciones más simples. Medio siglo después, el Teorema de Representación de Kolmogorov-Arnold ofreció una respuesta tan afirmativa como sorprendente: toda función continua de varias variables puede expresarse exactamente mediante sumas y composiciones de funciones de una sola variable. Sin embargo, este hito parecía condenado a quedar como una brillante curiosidad teórica. Los posteriores resultados de Vitushkin (que negaban poder exigir suavidad en las superposiciones) y de Sternfeld (que limitaban inferiormente la cantidad de términos necesarios) mostraron que esta representación exacta no era transferible a la práctica. Así permaneció el problema durante décadas, hasta que un giro reciente en la inteligencia artificial resucitó este profundo resultado matemático utilizándolo como plano arquitectónico para una nueva generación de modelos: las redes de Kolmogorov-Arnold o KANs (por sus siglas en inglés, *Kolmogorov-Arnold Networks*).

En la primera parte de este trabajo, presentamos una demostración completa y autocontenida del Teorema de Representación de Kolmogorov-Arnold (Teorema 1.9) para el caso general. Nuestra aportación principal consiste en exponer este denso resultado de un modo didáctico a través de una formulación matemática original: un nuevo operador de composición aplicable a matrices de funciones (el operador $\star$ de la Definición 1.2). Esta herramienta no solo simplifica la demostración, sino que constituye la base estructural sobre la que se articula el resto del trabajo.

A continuación, exploramos las limitaciones de este teorema a partir de los resultados de Sternfeld (sobre la dimensionalidad de las matrices, Teorema 2.5) y Vitushkin (sobre la suavidad de las funciones, Teorema 2.8). El valor de esta sección radica en la síntesis y unificación de una bibliografía muy dispersa para explicar por qué la representación matemática exacta es inviable en la práctica. De este modo, se establece la justificación teórica fundamental que motiva el salto hacia las aproximaciones prácticas de la siguiente sección.

Posteriormente, abordamos la teoría y arquitectura de las KANs. Planteamos el desarrollo de manera progresiva, lo que permite su plena comprensión sin que sea indispensable poseer conocimientos previos avanzados en *Machine Learning*. Además de detallar el funcionamiento estándar de la red, proponemos y justificamos modificaciones originales diseñadas para optimizar su entrenamiento y expresividad. Entre estos añadidos destacan la implementación de capas heterogéneas, la capacidad de combinar múltiples familias de funciones en una misma arista (Definición 3.7), la inclusión de un algoritmo de selección automática de familias funcionales (Subsección 3.7.2), la modificación de la poda original (Subsección 3.7.1) y un reescalado dinámico para mejorar las aristas (Subsección 3.3.3).

Para materializar este marco teórico, explicamos cómo el uso del ecosistema JAX y la vectorización de las evaluaciones permiten integrar nuestras contribuciones. Como resultado, hemos creado un repositorio público (Repositorio del proyecto) en el que implementamos las KANs tal y como las describimos, incluyendo una interfaz para interactuar con los modelos.

Finalmente, validamos nuestra implementación y aportaciones mediante una serie de experimentos. Examinamos y comparamos el rendimiento de las diferentes familias funcionales en distintos escenarios y arquitecturas. Asimismo, evaluamos la eficacia de nuestros métodos de reescalado y analizamos cómo el algoritmo de poda y la selección de familias contribuyen a destilar la arquitectura óptima de la red. Además, enfrentamos nuestra biblioteca a la implementación original propuesta en Liu et al. [2024b], realizando una comparativa directa para constatar las mejoras obtenidas tanto en la calidad y precisión de los resultados como en la velocidad de entrenamiento y eficiencia computacional.

Capítulo 1

Teorema de Representación de Kolmogorov-Arnold

1.1. Preliminares

Asumiremos que todas las funciones son continuas salvo que indiquemos lo contrario, y que todo subconjunto llamado $\mathbb{D} \subset \mathbb{R}^n$ es compacto. Por otra parte, en lo que sigue, las funciones de una variable dependerán de x y las de varias de $\mathbf{x}$ (en negrita).

Definición 1.1 (Representabilidad simple)**.** Sea $f : \mathbb{D} \subset \mathbb{R}^n \to \mathbb{R}^m$ una función. Decimos que f es **simplemente representable** si y solo si existen funciones continuas $a_{i,j}(t) : \mathbb{R} \to \mathbb{R}$, con $1 \leq i \leq m$ y $1 \leq j \leq n$, tales que para cualquier $\mathbf{x} = (x_1, \ldots, x_n) \in \mathbb{D}$ se cumple

$$f(\mathbf{x}) = \left(\sum_{j=1}^{n} a_{1,j}(x_j), \sum_{j=1}^{n} a_{2,j}(x_j), \ldots, \sum_{j=1}^{n} a_{m,j}(x_j) \right).$$

Podemos describir este tipo de funciones mediante un operador.

Definición 1.2 (Operador $\star$)**.** Sean $\mathcal{M}_{m\times n}(\mathcal{C}(\mathbb{R}))$ el conjunto de matrices de dimensiones $m \times n$ cuyas entradas son funciones y $A(t) \in \mathcal{M}_{m\times n}(\mathcal{C}(\mathbb{R}))$ de la forma

$$A(t) = (a_{i,j}(t))_{1\leq i\leq m, 1\leq j\leq n}.$$

Definimos el **operador** $\star$ como la aplicación

$$\begin{aligned} \star : \mathcal{M}_{m\times n}(\mathcal{C}(\mathbb{R})) \times \mathbb{R}^n &\longrightarrow \mathbb{R}^m \\ (A(t), \mathbf{x}) &\longmapsto A(t) \star \mathbf{x}, \end{aligned}$$

en la que el vector resultante se define componente a componente como:

$$\begin{pmatrix} a_{1,1}(t) & \cdots & a_{1,n}(t) \\ \vdots & \ddots & \vdots \\ a_{m,1}(t) & \cdots & a_{m,n}(t) \end{pmatrix} \star \begin{pmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{pmatrix} = \begin{pmatrix} a_{1,1}(x_1) + a_{1,2}(x_2) + \cdots + a_{1,n}(x_n) \\ a_{2,1}(x_1) + a_{2,2}(x_2) + \cdots + a_{2,n}(x_n) \\ \vdots \\ a_{m,1}(x_1) + a_{m,2}(x_2) + \cdots + a_{m,n}(x_n) \end{pmatrix}.$$

Observemos que este operador actúa como el producto entre matrices y vectores, pero evaluando funciones en lugar de multiplicando escalares. De ahora en adelante las funciones de una variable contenidas en matrices estarán escritas en función de t, y toda matriz dependiente de t se asumirá que es una matriz de funciones.

Definición 1.3 (Operador $\star$ para cadenas de matrices)**.** Sean $(A_i(t))_{i=1}^{k}$ matrices tales que cada $A_i(t) \in \mathcal{M}_{d_i\times d_{i-1}}(\mathcal{C}(\mathbb{R}))$. Definimos la aplicación anidada del **operador** $\star$ **entre** dichas matrices como

$$A_k(t) \star A_{k-1}(t) \star \cdots \star A_1(t) \star \mathbf{x} = A_k(t) \star \big(A_{k-1}(t) \star (\cdots \star (A_1(t) \star \mathbf{x}))\big),$$

para todo $\mathbf{x} \in \mathbb{D}$.

Estas son algunas propiedades del operador $\star$ que utilizaremos:

- **Compatibilidad de dimensiones:** $B(t) \star A(t) \star \mathbf{x}$ está bien definido si y solo si el número de columnas de $B(t)$ coincide con el número de filas de $A(t)$. Esto se extiende a cadenas de matrices compuestas por $\star$ como en la definición anterior.
- **Restricción del dominio:** el dominio de las funciones que componen la matriz $B(t)$ depende de la imagen de las funciones de $A(t)$. Específicamente, cada $b_{i,j}(t)$ de $B(t)$ debe estar definida en un intervalo que cubra la imagen de la suma de las funciones de la fila j de $A(t)$.
- **No conmutatividad:** el operador $\star$ no conmuta. Basta con tomar $A(t) = (\cos(t))$ y $B(t) = (\sin(t))$:

$$A(t) \star B(t) \star x = (\cos(\sin(x))) \quad \not\equiv \quad B(t) \star A(t) \star x = (\sin(\cos(x))).$$

Proposición 1.4. *Una función $f : \mathbb{D} \subset \mathbb{R}^n \to \mathbb{R}^m$ es **simplemente representable** si y solo si existe una matriz $A(t) \in \mathcal{M}_{m\times n}(\mathcal{C}(\mathbb{R}))$ tal que para todo $\mathbf{x} \in \mathbb{D}$*

$$f(\mathbf{x}) = A(t) \star \mathbf{x}.$$

Veamos un par de ejemplos.

Ejemplo 1.5 (Ejemplo elemental de función simplemente representable)**.** Consideremos la función $f : \mathbb{R}^2 \to \mathbb{R}^2$ con $f(x, y) = (x + y, x - y)$. Se puede comprobar que es simplemente representable dado que

$$\begin{pmatrix} t & t \\ t & -t \end{pmatrix} \star \begin{pmatrix} x \\ y \end{pmatrix} = \begin{pmatrix} x+y \\ x-y \end{pmatrix}.$$

Ejemplo 1.6 (Función no simplemente representable)**.** Sea $f(x, y) = xy$. Para ver que no es simplemente representable, basta con ver que no existen funciones continuas $a_{1,1}(t)$ y $a_{1,2}(t)$ tales que se cumpla la igualdad

$$xy = \begin{pmatrix} a_{1,1}(t) & a_{1,2}(t) \end{pmatrix} \star \begin{pmatrix} x \\ y \end{pmatrix} = a_{1,1}(x) + a_{1,2}(y).$$

Supongamos que dichas funciones existen. Si fijamos $y = 0$ obtenemos $a_{1,1}(x) = -a_{1,2}(0)$, de modo que $a_{1,1}(x)$ es constante. Por otro lado, fijando $y = 1$ se llega a que $a_{1,1}(x) = x - a_{1,2}(1)$, que es afín, es decir, no constante. Esto contradice lo que acabamos de decir, de modo que no pueden existir estas funciones.

Definición 1.7. Sea $f : \mathbb{D} \subset \mathbb{R}^n \to \mathbb{R}^m$. Entonces decimos que f es **representable** si y solo si existen matrices $(A_i(t))_{i=1}^N$ de dimensiones compatibles tales que

$$f(\mathbf{x}) = A_N(t) \star \cdots \star A_1(t) \star \mathbf{x}.$$

Pasemos a comprobar que existen funciones representables que no son simplemente representables.

Ejemplo 1.8 (Función representable no simplemente representable)**.** Consideremos la función del Ejemplo 1.6 que sabemos que no es simplemente representable. Veamos que es representable tomando las matrices

$$B(t) = \begin{pmatrix} \frac{t}{2} & \frac{t}{2} \\ \frac{t}{2} & -\frac{t}{2} \end{pmatrix}, \qquad A(t) = \begin{pmatrix} t^2 & -t^2 \end{pmatrix}.$$

Calculemos $A(t) \star B(t) \star \begin{pmatrix} x \\ y \end{pmatrix}$.

- $B(t) \star \begin{pmatrix} x \\ y \end{pmatrix} = \begin{pmatrix} \frac{t}{2} & \frac{t}{2} \\ \frac{t}{2} & -\frac{t}{2} \end{pmatrix} \star \begin{pmatrix} x \\ y \end{pmatrix} = \begin{pmatrix} \frac{x+y}{2} \\ \frac{x-y}{2} \end{pmatrix}$

- $A(t) \star \left(B(t) \star \begin{pmatrix} x \\ y \end{pmatrix} \right) = \begin{pmatrix} t^2 & -t^2 \end{pmatrix} \star \begin{pmatrix} \frac{x+y}{2} \\ \frac{x-y}{2} \end{pmatrix} = \left(\frac{x+y}{2}\right)^2 - \left(\frac{x-y}{2}\right)^2 = xy.$

Enunciamos el Teorema de Representación de Kolmogorov-Arnold en su formulación clásica.

Teorema 1.9 (Teorema de Representación de Kolmogorov-Arnold)**.** *Sea $f : \mathbb{D} \subset \mathbb{R}^n \to \mathbb{R}$ una función continua. Entonces existen funciones continuas de una variable $\Phi_i : \mathbb{R} \to \mathbb{R}$ y $\phi_{i,j} : \mathbb{R} \to \mathbb{R}$, para $1 \leq i \leq 2n+1$ y $1 \leq j \leq n$, tales que para cualquier $\mathbf{x} = (x_1, \ldots, x_n) \in \mathbb{D}$ se cumple que*

$$f(\mathbf{x}) = \sum_{i=1}^{2n+1} \Phi_i \left(\sum_{j=1}^{n} \phi_{i,j}(x_j) \right).$$

Este teorema puede ser traducido a nuestra notación del siguiente modo:

Teorema 1.10 (Teorema de Representación de Kolmogorov-Arnold)**.** *Sea $f : \mathbb{D} \subset \mathbb{R}^n \to \mathbb{R}$. Entonces existen dos matrices $\Gamma(t)$ y $\Theta(t)$ de dimensiones $1 \times (2n+1)$ y $(2n+1) \times n$ respectivamente tales que*

$$f(\mathbf{x}) = \Gamma(t) \star \Theta(t) \star \mathbf{x}.$$

De ahora en adelante asumiremos que $(\kappa_i)_{i=1}^n \subset \mathbb{R} \setminus \mathbb{Q}$ es una sucesión fija que cumple la Definición A.5, es decir, que es linealmente independiente sobre $\mathbb{Q}$. Sabemos que existe al menos una gracias a la Proposición A.6, cuyos términos son positivos, así que asumiremos que para todo $1 \leq i \leq n$ se tiene que $\kappa_i > 0$.

Definición 1.11. Definimos $\Gamma_\gamma(t) \in \mathcal{M}_{m \times n}(\mathcal{C}(\mathbb{R}))$ como la matriz cuyas entradas son todas idénticamente la función $\gamma(t)$. Es decir,

$$\Gamma_\gamma(t) = (\gamma(t))_{1 \leq i \leq m,\, 1 \leq j \leq n}.$$

No conoceremos, por lo general, el dominio concreto de $\gamma(t)$. Sin embargo, siempre estará contenido en un compacto. Por otro lado, decimos que una matriz $(\kappa_j \theta_i(t))_{i,j} = (a_{i,j}(t))_{1 \leq i \leq m,\, 1 \leq j \leq n}$ es de **Kolmogorov** si sus entradas tienen la forma $a_{i,j}(t) = \kappa_j \theta_i(t)$. En este caso el dominio de las funciones siempre será $[0, 1]$.

Teorema 1.12 (Forma de Kolmogorov)**.** *Sea $f : \mathbb{D} \subset \mathbb{R}^n \to \mathbb{R}$ una función. Entonces, existen dos matrices $\Gamma_\gamma(t)$ y $\Theta(t)$ de dimensiones $1 \times (2n+1)$ y $(2n+1) \times n$ respectivamente, con $\Theta(t)$ de Kolmogorov, tales que:*

$$f(\mathbf{x}) = \Gamma_\gamma(t) \star \Theta(t) \star \mathbf{x}.$$

1.2. Demostración del Teorema

Para esta demostración utilizaremos como referencia el artículo Dzhenzher and Skopenkov [2021]. Lo adaptaremos para cubrir el caso general en lugar de $n = 2$, y lo formularemos utilizando la notación introducida en la sección anterior. Definimos $\|\cdot\|_\infty$ como la norma del supremo sobre funciones que van a $\mathbb{R}$. En el caso de una función vectorial, es decir, $g(\mathbf{x}) = (g_1(\mathbf{x}), \ldots, g_n(\mathbf{x})) \in \mathbb{R}^n$, definimos $\|g\|_\infty = \max_{1 \leq k \leq n} \|g_k\|_\infty$. Esta norma puede ser extendida de manera natural **a matrices de funciones** tomando el máximo de todas las entradas. Demostraremos el teorema para el caso $\mathbb{D} = I^n = [0, 1]^n$. La extensión del mismo a cualquier compacto se puede encontrar en el trabajo Ostrand [1965].

1.2.1. Funciones de separación racional

Definición 1.13 (Función de separación racional)**.** Sean $\varphi : I^1 \to \mathbb{R}$ y $(J_k)_{k=1}^N$ una sucesión de intervalos cerrados disjuntos en I^1. Se dice que φ **separa racionalmente los intervalos** J_k si y solo si cumple las siguientes dos propiedades:

1. φ toma valores racionales sobre cada uno de los intervalos.

2. Dados $x_p \in J_p$ y $x_q \in J_q$, se cumple que $\varphi(x_p) = \varphi(x_q)$ si y solo si $p = q$.

Definición 1.14. Sean $n, N \in \mathbb{N}^*$, $1 \leq k \leq 2n+1$ y $Z_k(N) = [-1, \frac{N-k}{2n+1}] \cap \mathbb{Z}$. Dado un $j \in Z_k(N)$, definimos el **extremo inferior** $l_{j,k}(N) = \frac{(2n+1)j+k}{N}$ y el **extremo superior** $u_{j,k}(N) = l_{j,k}(N) + \frac{2n}{N}$, los cuales definen el intervalo $I_{j,k}(N) = [l_{j,k}(N), u_{j,k}(N)]$. Además, denotamos el **extremo inferior mayor o igual que 0** como $l_{j,k,0}(N) = \text{máx}(0, l_{j,k}(N))$. Fijado k, llamamos **capa** $\boldsymbol{k}$ a la unión de los intervalos $\{I_{j,k}(N)\}_{j \in Z_k(N)}$ y **malla** $\boldsymbol{k}$ a la unión de los hipercubos $C = I_{j_1,k}(N) \times \cdots \times I_{j_n,k}(N)$ tales que $j_1, \ldots, j_n \in Z_k(N)$.

La variable k sirve para indexar las capas y las mallas. Siempre que estemos trabajando con n, entonces habrá $2n+1$ capas, es decir, $1 \leq k \leq 2n+1$. Por otro lado, resulta útil notar que la capa k es la capa 1 desplazada $\frac{k-1}{N}$ unidades a la derecha.

Estructura de las capas

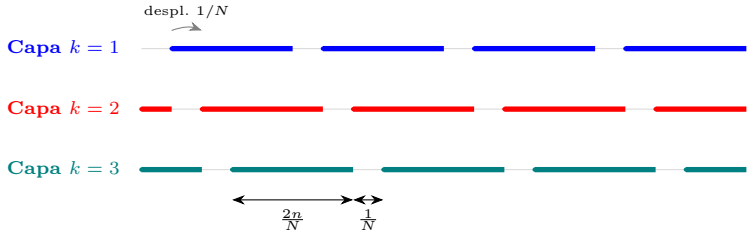

Figura 1.1: Representación de las capas para $N = 20$ y $k = 1, 2, 3$.

Proposición 1.15 (Cobertura de la malla)**.** *Sean* $\mathbf{x} \in I^n$ *y* $N \in \mathbb{N}^*$*. Entonces existen por lo menos* $n+1$ *mallas construidas con* N *que contienen a* $\mathbf{x}$*.*

Demostración. Consideremos $\mathbf{x} = (x_1, \ldots, x_n) \in I^n$. Sabemos que $\mathbf{x}$ está en la malla k si y solo si existen $(j_i)_{i=1}^n \subset Z_k(N)$ tales que $\mathbf{x} \in I_{j_1,k}(N) \times I_{j_2,k}(N) \times \cdots \times I_{j_n,k}(N)$, y esto es equivalente a pedir que $Nx_i \in [(2n+1)j_i + k, (2n+1)j_i + k + 2n]$ para todo $1 \leq i \leq n$. Basta con ver que se cumple esto para al menos $n+1$ valores de k distintos. Tomemos la coordenada i-ésima de $\mathbf{x}$, x_i: dividimos $\lfloor Nx_i \rfloor$ entre $2n+1$,

$$\lfloor Nx_i \rfloor = (2n+1) \left\lfloor \frac{Nx_i}{2n+1} \right\rfloor + r,$$

con $0 \leq r \leq 2n$ el residuo y $m = \left\lfloor \frac{Nx_i}{2n+1} \right\rfloor$ el cociente. Tenemos que para cualquier $0 \leq s \leq 2n-1$ entero:

- $(2n+1)m + (r-s) \leq (2n+1)m + r \leq Nx_i$.
- $Nx_i < (2n+1)m + r + 1 \leq (2n+1)m + r + (2n-s) = (2n+1)m + (r-s) + 2n$.

Con lo cual se cumple que si $K = r - s$, entonces $Nx_i \in [(2n+1)m + K, (2n+1)m + K + 2n]$. Dado que r es fijo y $0 \leq s \leq 2n-1$ puede tomar $2n$ valores distintos, existen $2n$ valores posibles para K. Si $K > 0$, esto corresponde a la capa de índice K. Si no, podemos reescribir el intervalo de la siguiente forma,

$$[(2n+1)m+K,\ (2n+1)m+K+2n] = [(2n+1)(m-1)+(K+2n+1),\ (2n+1)(m-1)+(K+2n+1)+2n],$$

con lo cual el intervalo corresponde a la capa $K' = K+2n+1$. Sabemos que $K = r-s \in [r-(2n-1), r]$, de modo que

- **Si** $K > 0$**:** el índice de la capa es el propio K. El mayor índice posible es r (que ocurre con $s = 0$) y el menor es 1 (con $s = r-1$).

- **Si $K \leq 0$:** el índice de la capa es $K' = K + 2n + 1$. El menor valor que puede tomar K es $r - (2n-1)$ (con $s = 2n-1$), lo que corresponde al índice de capa mínimo, $K' = r + 2$. El mayor valor posible es $K = 0$ (con $s = r$), que corresponde al índice de capa máximo, $K' = 2n + 1$.

En ambos escenarios el índice de la capa es válido (está entre 1 y $2n+1$), y dado que las capas del primer caso son todas distintas de las del segundo, hay una biyección entre los $2n$ posibles valores de K y sus correspondientes índices. Consecuentemente, x_i cae en al menos $2n$ capas de intervalos. Este razonamiento lo podemos aplicar para cada una de las coordenadas de $\mathbf{x} = (x_1, \ldots, x_n)$. Puesto que cada coordenada solo puede dejar de pertenecer a una de las $2n+1$ capas totales, por el principio del palomar, todas deben coincidir como mínimo en $(2n+1) - n = n + 1$ capas. □

Pasemos a un resultado que nos dirá cómo podemos utilizar las funciones de separación racional. Aclaramos que dada una función $\Omega : I^1 \to \mathbb{R}$ constante en un intervalo $J \subset I^1$, escribiremos $\Omega(J)$ para denotar el valor que toma la función en J, no el conjunto imagen.

Superposición de mallas (cobertura $n+1$)

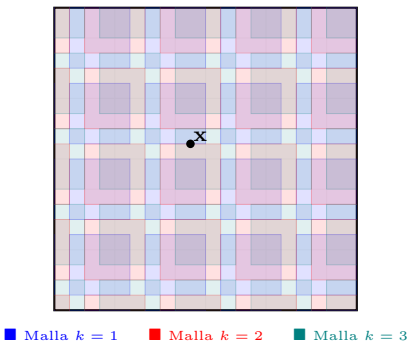

Figura 1.2: Intersección de las mallas $k = 1, 2, 3$ con $N = 20$.

Lema 1.16 (Aproximación de funciones escalares de una variable por funciones de separación racional)**.** *Sean $\psi : I^1 \to \mathbb{R}$, $1 \leq k \leq 2n+1$ y $\varepsilon > 0$. Entonces existe un $N_0(\psi, \varepsilon) > 0$ tal que, para todo $N > N_0$ y para cualquier subconjunto $G \subset \mathbb{Q}$ finito, existe $\varphi : I^1 \to \mathbb{R}$ tal que:*

1. *$\|\psi - \varphi\|_\infty < \varepsilon$.*
2. *φ separa racionalmente la capa k construida con N.*
3. *Para todo x en la capa k, $\varphi(x) \notin G$.*

Demostración. Sabemos que ψ es una función uniformemente continua por ser I^1 compacto. Consecuentemente existe un $\delta > 0$ tal que si $|x - y| < \delta$ entonces $|\psi(x) - \psi(y)| < \frac{\varepsilon}{2}$. Definimos $N_0 = \lceil \frac{2n+1}{\delta} \rceil$ y tomamos $N > N_0$. Construiremos φ exigiendo unas ciertas propiedades y comprobaremos que es posible satisfacerlas y que son suficientes:

- Para cada $j \in Z_k(N)$ escogemos $c_{j,k} \in \mathbb{Q} \setminus G$ tal que $|\psi(l_{j,k,0}(N)) - c_{j,k}| < \frac{\varepsilon}{2}$ (podemos hacer esto por ser $\mathbb{Q}$ denso y G finito). Definimos entonces $\varphi(t) = c_{j,k}$ para todo $t \in I_{j,k} \cap I^1$.
- Forzamos que no haya valores repetidos entre intervalos, es decir, que si $i \neq j$ entonces $c_{i,k} \neq c_{j,k}$.
- En cada hueco ubicado entre dos intervalos, definimos φ de manera que sea una función lineal que interpole los extremos de cada uno, lo que hace a la función continua. No es necesario considerar los huecos exteriores, puesto que no hay (Proposición A.3).

Que φ cumple las dos últimas condiciones es una consecuencia inmediata de lo anteriormente dicho. Para la primera, vemos que si $x_0 \in I^1$ entonces $|\psi(x_0) - \varphi(x_0)| < \varepsilon$. Distinguimos dos casos:

- **Caso x_0 en la capa k**: sea j_0 tal que $x_0 \in I_{j_0,k}(N)$. Dado que $|x_0 - l_{j_0,k,0}(N)| \leq \frac{2n}{N} < \delta$, entonces $|\psi(x_0) - \psi(l_{j_0,k,0}(N))| < \frac{\varepsilon}{2}$. Con lo cual

$$\begin{aligned}|\psi(x_0) - \varphi(x_0)| &\leq |\psi(x_0) - \psi(l_{j_0,k,0}(N))| + |\psi(l_{j_0,k,0}(N)) - \varphi(x_0)| \\ &= |\psi(x_0) - \psi(l_{j_0,k,0}(N))| + |\psi(l_{j_0,k,0}(N)) - \varphi(l_{j_0,k,0}(N))| \\ &< \frac{\varepsilon}{2} + \frac{\varepsilon}{2} = \varepsilon.\end{aligned}$$

- **Caso x_0 entre dos intervalos**: sea $j_0 \in Z_k(N)$ el índice del intervalo inmediatamente anterior a x_0. Como todos los huecos interiores miden $\frac{1}{N}$, entonces $u_{j_0,k} < x_0 < l_{j_0+1,k}$ y $|x_0 - l_{j_0+1,k}(N)| < \frac{1}{N}$. De esto se sigue que para algún $\alpha \in (0,1)$ $x_0 = l_{j_0+1,k}(N) - \frac{\alpha}{N}$. Dicho α además permite construir x_0 como combinación lineal de $u_{j_0,k}$ y $l_{j_0+1,k}$

$$x_0 = \alpha\left(l_{j_0+1,k}(N) - \frac{1}{N}\right) + (1-\alpha)l_{j_0+1,k}(N) = \alpha u_{j_0,k}(N) + (1-\alpha)l_{j_0+1,k}(N).$$

  Por construcción φ interpola linealmente los puntos $(u_{j_0,k}(N), \varphi(u_{j_0,k}(N)))$ y $(l_{j_0+1,k}(N), \varphi(l_{j_0+1,k}(N)))$, así que podemos aplicar la propiedad de linealidad directamente. Consecuentemente, utilizando que el único intervalo que puede intersecar $\mathbb{R}_{\leq 0}$ es el primero y que por lo tanto $0 \leq u_{j_0,k}$ y $0 < l_{j_0+1,k}$, deducimos que

$$\varphi(x_0) = \alpha\varphi(u_{j_0,k}(N)) + (1-\alpha)\varphi(l_{j_0+1,k}(N)) = \alpha\varphi(l_{j_0,k,0}(N)) + (1-\alpha)\varphi(l_{j_0+1,k,0}(N)),$$

  donde hemos utilizado que $\varphi(u_{j_0,k}(N)) = \varphi(l_{j_0,k,0}(N))$ por ser ambos extremos del mismo intervalo. Con lo cual:

$$\begin{aligned}|\psi(x_0) - \varphi(x_0)| &= \Big|\psi(x_0)(\alpha + (1-\alpha)) - \big(\alpha\varphi(l_{j_0,k,0}(N)) + (1-\alpha)\varphi(l_{j_0+1,k,0}(N))\big)\Big| \\ &\leq \alpha\big|\psi(x_0) - \varphi(l_{j_0,k,0}(N))\big| + (1-\alpha)\big|\psi(x_0) - \varphi(l_{j_0+1,k,0}(N))\big|.\end{aligned}$$

  Veamos que ambos términos son menores que ε y habremos terminado.

  1. Dado que $|x_0 - l_{j_0,k,0}(N)| \leq |x_0 - l_{j_0,k}(N)| \leq \frac{2n}{N} + \frac{1}{N} < \delta$, concluimos que:

$$\begin{aligned}&\big|\psi(x_0) - \varphi(l_{j_0,k,0}(N))\big| \leq |\psi(x_0) - \psi(l_{j_0,k,0}(N))| + |\psi(l_{j_0,k,0}(N)) - \varphi(l_{j_0,k,0}(N))| \\ &< \frac{\varepsilon}{2} + \frac{\varepsilon}{2} = \varepsilon.\end{aligned}$$

  2. Lógicamente tenemos que $\big|x_0 - l_{j_0+1,k,0}(N)\big| \leq \frac{1}{N} < \delta$. Así que:

$$\begin{aligned}\big|\psi(x_0) - \varphi(l_{j_0+1,k,0}(N))\big| &\leq \big|\psi(x_0) - \psi(l_{j_0+1,k,0}(N))\big| \\ &\quad + \big|\psi(l_{j_0+1,k,0}(N)) - \varphi(l_{j_0+1,k,0}(N))\big| \\ &< \frac{\varepsilon}{2} + \frac{\varepsilon}{2} = \varepsilon.\end{aligned}$$

  De modo que queda concluida la demostración.

□

En el Ejemplo A.4 se puede ver cómo se aplica el anterior resultado para el caso $\psi(x) = x^2$, $k = 1$ y $\varepsilon = \frac{1}{2}$. Ahora extenderemos el Lema 1.16 a funciones de $\mathbb{R}$ a $\mathbb{R}^n$.

Lema 1.17 (Aproximación de funciones vectoriales por funciones de separación racional)**.** *Sean $\psi : I^1 \to \mathbb{R}^{2n+1}$ y $\varepsilon > 0$. Entonces existe $N_0 > 0$ tal que para cualquier $N > N_0$ existe $\varphi = (\varphi_1, \ldots, \varphi_{2n+1}) : I^1 \to \mathbb{R}^{2n+1}$ que cumple las siguientes tres condiciones:*

1. $\|\psi - \varphi\|_\infty < \varepsilon$.
2. *φ_k separa racionalmente la capa k.*

3. *Los valores* $\varphi_k(I_{j,k})$ *son distintos entre sí (para todo* $1 \le k \le 2n+1$ *y* $j \in Z_k(N)$*).*

Demostración. Sea $\psi = (\psi_1, \dots, \psi_{2n+1})$. Aplicando el Lema 1.16 a cada componente ψ_k, obtenemos los correspondientes valores $N_{0,k}$. Definimos $N_0 = \max\limits_{1\le k\le 2n+1} N_{0,k}$ y fijamos $N > N_0$.

Construiremos φ de forma inductiva:

1. **Base ($k = 1$):** Aplicamos el Lema 1.16 a la función ψ_1 con el conjunto $G_1 = \emptyset$ y ε. Obtenemos una función φ_1 que separa racionalmente la capa 1. Definimos el conjunto de valores alcanzados: $H_1 = \{\varphi_1(I_{j,1}) : j \in Z_1(N)\}$.

2. **Paso inductivo ($k = p$):** Supongamos que hemos construido $\varphi_1, \dots, \varphi_{p-1}$ y sus correspondientes conjuntos de valores $H_1, \dots, H_{p-1}$. Para construir φ_p, aplicamos el Lema 1.16 a ψ_p con ε, N y el conjunto de valores prohibidos

$$G_p = H_1 \cup H_2 \cup \cdots \cup H_{p-1},$$

que es claramente finito. Nótese que podemos utilizar en todos los casos el mismo N dado que en el Lema 1.16 N_0 no depende de G. Obtenemos así φ_p, que separa racionalmente la capa p y cuyos valores no pertenecen a G_p, garantizando que $\varphi_p(I_{j,p}) \neq \varphi_i(I_{j',i})$ para todo $i < p$. Definimos $H_p = \{\varphi_p(I_{j,p}) : j \in Z_p(N)\}$.

3. Tras $2n+1$ pasos, obtenemos la función $\varphi = (\varphi_1, \dots, \varphi_{2n+1})$.

Por construcción, se satisfacen las propiedades de separación racional y distinción de valores. Finalmente, tenemos que:

$$\|\psi - \varphi\|_\infty = \max_{1\le k\le 2n+1} \|\psi_k - \varphi_k\|_\infty < \varepsilon.$$

□

1.2.2. Teoremas de aproximación y estabilidad

En esta sección demostraremos los dos resultados más importantes para poder demostrar el Teorema 1.12.

Definición 1.18. Sea $f : I^n \to \mathbb{R}$. Decimos que una matriz $\Theta(t)$ de dimensiones $m \times n$ es **λ-preKolmogorov** para f si y solo si existe una matriz $\Gamma_\gamma(t)$ de dimensiones $1 \times m$ tal que:

1. $\|\gamma\|_\infty \le \|f\|_\infty$.
2. $\|f(\cdot) - \Gamma_\gamma(t) \star \Theta(t) \star (\cdot)\|_\infty < \lambda \|f\|_\infty$.

En caso de tener $f \equiv 0$ decimos, por convención, que cualquier matriz de funciones compatible es λ-preKolmogorov.

Teorema 1.19 (Teorema de aproximación)**.** *Sean* $f : I^n \to \mathbb{R}$, $\Theta(t) = (\kappa_j \theta_i(t))_{i,j}$ *una matriz de dimensiones* $(2n+1) \times n$ *de Kolmogorov y* $\varepsilon > 0$*. Entonces, existe una matriz* $\Theta'(t)$ *de dimensiones* $(2n+1) \times n$ *de Kolmogorov que cumple las siguientes dos propiedades:*

- $\|\Theta(t) - \Theta'(t)\|_\infty < \varepsilon$.
- $\Theta'(t)$ *es* $\left(\frac{2n+2}{2n+3}\right)$*-preKolmogorov para* f*.*

Demostración. Si $f \equiv 0$, cualquier matriz es $\left(\frac{2n+2}{2n+3}\right)$-preKolmogorov, de modo que nos quedamos con $\Theta(t)$ y hemos terminado. Supongamos pues que $\|f\|_\infty > 0$. Definimos $\psi(x) = (\theta_1(x), \dots, \theta_{2n+1}(x))$. Utilizamos el Lema 1.17 con ψ y $\varepsilon' = \frac{\varepsilon}{\max_{1\le i\le n} \kappa_i}$ para construir un entero N_0. Por ser f uniformemente continua sabemos que hay un $\delta > 0$ tal que para $\mathbf{x}, \mathbf{x}'$ con $\|\mathbf{x} - \mathbf{x}'\|_\infty < \delta$ se tiene que $|f(\mathbf{x}) - f(\mathbf{x}')| < \frac{1}{2n+2}\|f\|_\infty$. Escogemos pues $N > \max(N_0, \lceil \frac{2n}{\delta} \rceil)$. Mediante este entero y habiendo aplicado el Lema 1.17 obtenemos una aplicación $\varphi(x) = (\varphi_1(x), \dots, \varphi_{2n+1}(x))$ con las propiedades que garantiza dicho Lema (para el N escogido, y las mallas asociadas según la Definición 1.14). Para simplificar la notación, nos referiremos a los objetos de la Definición 1.14 omitiendo la dependencia del parámetro N, dado que este permanecerá fijo durante el resto de la demostración. Definimos entonces el conjunto de funciones $\Phi_k(\mathbf{x}) = \sum_{i=1}^n \kappa_i \varphi_k(x_i)$ para $1 \le k \le 2n+1$.

- Pasamos a ver que toda función $\Phi_k(\mathbf{x})$ toma valores constantes y distintos sobre los hipercubos de la malla k. Lo primero es claro, puesto que como φ_k es constante en cada intervalo de la capa k, la suma Φ_k es constante en cada producto de los mismos, es decir, en cada hipercubo. Para ver que son distintos, consideramos la malla k_0 y dos puntos $\mathbf{x}$ y $\mathbf{x}'$ pertenecientes a dos hipercubos de dicha malla. Supongamos pues que

$$\Phi_{k_0}(\mathbf{x}) = \Phi_{k_0}(\mathbf{x}').$$

  Entonces,

$$0 = \Phi_{k_0}(\mathbf{x}) - \Phi_{k_0}(\mathbf{x}') = \sum_{i=1}^{n} \kappa_i\big(\varphi_{k_0}(x_i) - \varphi_{k_0}(x_i')\big).$$

  Por la Definición A.5, por ser $\Theta(t)$ de Kolmogorov y por la construcción de las funciones φ_i tenemos que

$$\varphi_{k_0}(x_i) = \varphi_{k_0}(x_i') \qquad \text{para todo } 1 \leq i \leq n.$$

  El Lema 1.17 garantiza que los valores de φ_{k_0} en cada intervalo de la capa k_0 son distintos, de modo que necesariamente x_i y x_i' pertenecen al mismo intervalo para todo i. Por tanto, $\mathbf{x}$ y $\mathbf{x}'$ están en el mismo hipercubo de la malla k_0.
- Ahora veremos que, dados dos hipercubos C y C' en las mallas k y k', respectivamente, si los valores constantes de Φ_k y $\Phi_{k'}$ sobre ellos coinciden, entonces necesariamente $C = C'$. Supongamos que

$$C = I_{j_1,k} \times \cdots \times I_{j_n,k} \qquad \text{y} \qquad C' = I_{j_1',k'} \times \cdots \times I_{j_n',k'}.$$

  Entonces,

$$\Phi_k(C) = \Phi_{k'}(C') \iff \sum_{i=1}^{n} \kappa_i\big(\varphi_k(I_{j_i,k}) - \varphi_{k'}(I_{j_i',k'})\big) = 0.$$

  Si $k \neq k'$, por la construcción utilizada en el Lema 1.17, los conjuntos de valores racionales que toman φ_k y $\varphi_{k'}$ sobre los intervalos son disjuntos. Sin embargo, razonando análogamente al punto anterior tenemos que

$$\varphi_k(I_{j_i,k}) = \varphi_{k'}(I_{j_i',k'}) \quad \text{para todo } i,$$

  lo cual es imposible por construcción. Por tanto $k = k'$ y por el punto anterior $C = C'$.

Ahora construimos la matriz que buscamos:

$$\Theta'(t) = (\kappa_j \varphi_i(t))_{i,j}.$$

Veamos pues las dos condiciones que dicha matriz debe cumplir. Para ver la primera, basta con observar que

$$\|\Theta(t) - \Theta'(t)\|_\infty = \max_{i,j} \|\kappa_j\theta_i - \kappa_j\varphi_i\|_\infty = \max_{i,j} \kappa_j \|\theta_i - \varphi_i\|_\infty < \max_{i,j} \kappa_j \frac{\varepsilon}{\max_l \kappa_l} = \varepsilon.$$

Pasemos a ver que es $\frac{2n+2}{2n+3}$-preKolmogorov para f: consideremos una función γ lineal a trozos, interpoladora, constante en los huecos exteriores y tal que

$$\gamma(\Phi_k(I_{j_1,k} \times \cdots \times I_{j_n,k})) = \frac{1}{n+1} f(l_{j_1,k,0}, \ldots, l_{j_n,k,0})$$

para toda combinación de valores $(j_1, \ldots, j_n) \in Z_k(N)^n$ y $1 \leq k \leq 2n+1$ (nótese que podemos exigir esto ya que acabamos de ver que las funciones Φ_k toman valores distintos sobre todos los hipercubos).

- **Condición** $\|\gamma\|_\infty \leq \|f\|_\infty$: el valor absoluto de γ está acotado por $\|f\|_\infty$ por su construcción, así que $\|\gamma\|_\infty \leq \frac{1}{n+1}\|f\|_\infty < \|f\|_\infty$.

- **Condición** $\|f(\cdot)-\Gamma_\gamma(t)\star\Theta'(t)\star(\cdot)\|_\infty < \left(\frac{2n+2}{2n+3}\right)\|f\|_\infty$: recordamos que dados dos puntos $\mathbf{x}$,$\mathbf{x}'$ en un mismo hipercubo entonces $\|\mathbf{x}-\mathbf{x}'\|_\infty \leq \frac{2n}{N} < \delta$, y por tanto $|f(\mathbf{x})-f(\mathbf{x}')| < \frac{1}{2n+2}\|f\|_\infty$ por lo establecido en el inicio de la demostración. Por la Proposición 1.15 tenemos que $\mathbf{x}$ se encuentra en al menos $n+1$ hipercubos. Sean $\mathcal{M}_b$ un subconjunto del conjunto de mallas en el que $\mathbf{x}$ cae tal que $|\mathcal{M}_b| = n+1$ y $\mathcal{M}_m$ el complementario dentro del conjunto de las mallas, con $|\mathcal{M}_m| = n$. Consideramos el error $E(\mathbf{x}) = |\Gamma_\gamma(t)\star\Theta'(t)\star\mathbf{x} - f(\mathbf{x})|$. Entonces

$$E(\mathbf{x}) = |\Gamma_\gamma(t)\star\Theta'(t)\star\mathbf{x} - f(\mathbf{x})| = \left|\begin{pmatrix}\gamma(t) & \dots & \gamma(t)\end{pmatrix}\star\begin{pmatrix}\Phi_1(\mathbf{x})\\ \vdots \\ \Phi_{2n+1}(\mathbf{x})\end{pmatrix} - f(\mathbf{x})\right|$$

$$\leq \underbrace{\left|\sum_{i\in\mathcal{M}_b}\gamma\left(\Phi_i(\mathbf{x})\right) - f(\mathbf{x})\right|}_{E_b:=\text{ Error en mallas en las que cae }\mathbf{x}} + \underbrace{\left|\sum_{i\in\mathcal{M}_m}\gamma(\Phi_i(\mathbf{x}))\right|}_{E_m:=\text{ Error en mallas en las que no cae }\mathbf{x}}.$$

  - **Caso** $i\in\mathcal{M}_b$**:** como $\gamma(\Phi_i(\mathbf{x})) = \frac{1}{n+1}f(l_{j_1,i,0},\dots,l_{j_n,i,0})$, entonces

$$E_b = \left|\sum_{i\in\mathcal{M}_b}\tfrac{1}{n+1}\left(f(l_{j_1,i,0},\dots,l_{j_n,i,0}) - f(\mathbf{x})\right)\right| \leq \tfrac{1}{n+1}\sum_{i\in\mathcal{M}_b}|f(l_{j_1,i,0},\dots,l_{j_n,i,0}) - f(\mathbf{x})|$$
$$< \frac{|\mathcal{M}_b|\cdot\|f\|_\infty}{(n+1)(2n+2)} = \frac{\|f\|_\infty}{2n+2}.$$

  - **Caso** $i\in\mathcal{M}_m$**:** por definición $\|\gamma\|_\infty \leq \frac{\|f\|_\infty}{n+1}$, luego:

$$E_m \leq \sum_{i\in\mathcal{M}_m}\frac{\|f\|_\infty}{n+1} = \frac{n}{n+1}\|f\|_\infty.$$

  Utilizando ambas cotas llegamos a

$$E(\mathbf{x}) \leq \frac{\|f\|_\infty}{2n+2} + \|f\|_\infty\cdot\frac{n}{n+1} = \|f\|_\infty\left(\frac{1}{2n+2}+\frac{n}{n+1}\right) = \|f\|_\infty\frac{2n+1}{2n+2} < \frac{2n+2}{2n+3}\|f\|_\infty.$$

  Con esto concluye la demostración de la segunda condición.

□

Teorema 1.20 (Teorema de estabilidad)**.** *Sean $f: I^n\to\mathbb{R}$ y $\Theta(t)$ una matriz de dimensiones $(2n+1)\times n$ que además es $\left(\frac{2n+2}{2n+3}\right)$-preKolmogorov para f. Entonces existe $\varepsilon>0$ tal que para cualquier otra matriz $\Theta'(t)$ se cumple que si $\|\Theta(t)-\Theta'(t)\|_\infty<\varepsilon$ entonces $\Theta'(t)$ es $\left(\frac{2n+2}{2n+3}\right)$-preKolmogorov para f.*

Demostración. Si $f\equiv 0$ cualquier ε aplica. Supongamos pues $\|f\|_\infty > 0$. Sabemos que $\Theta(t) = (\theta_{i,j}(t))_{i,j}$ es $\left(\frac{2n+2}{2n+3}\right)$-preKolmogorov para f, así que existe una matriz $\Gamma_\gamma(t)$ que cumple la Definición 1.18. Definimos

$$\eta = \frac{1}{2n+1}\left(\frac{2n+2}{2n+3}\|f\|_\infty - \|f(\cdot)-\Gamma_\gamma(t)\star\Theta(t)\star(\cdot)\|_\infty\right) > 0.$$

Dado que γ es continua, es uniformemente continua sobre su dominio. Por ello existe $\delta>0$ tal que si $|x-x'|<\delta$ entonces $|\gamma(x)-\gamma(x')|<\eta$. Escogemos $\varepsilon = \frac{\delta}{n}$ y verificamos que cumple la propiedad requerida.
Sean $\Phi_l(\mathbf{x}) = \sum_{j=1}^n\theta_{l,j}(x_j)$ y $\Phi'_l(\mathbf{x}) = \sum_{j=1}^n\theta'_{l,j}(x_j)$ con $1\leq l\leq 2n+1$, entonces

$$|\Gamma_\gamma(t)\star\Theta(t)\star\mathbf{x} - \Gamma_\gamma(t)\star\Theta'(t)\star\mathbf{x}| = \left|\sum_{l=1}^{2n+1}\left(\gamma(\Phi_l(\mathbf{x})) - \gamma(\Phi'_l(\mathbf{x}))\right)\right|. \tag{1.1}$$

Se cumple que

$$|\Phi_l(\mathbf{x}) - \Phi'_l(\mathbf{x})| = \left|\sum_{j=1}^{n}(\theta_{l,j}(x_j) - \theta'_{l,j}(x_j))\right| \leq \sum_{j=1}^{n}|\theta_{l,j}(x_j) - \theta'_{l,j}(x_j)| \leq n\|\Theta(t) - \Theta'(t)\|_\infty < \delta.$$

De modo que la diferencia de los argumentos de γ en la ecuación (1.1) es menor que δ, así que $|\gamma(\Phi_l(\mathbf{x})) - \gamma(\Phi'_l(\mathbf{x}))| < \eta$. Sumando sobre las $2n+1$ componentes de la representación, obtenemos

$$\sum_{l=1}^{2n+1}|\gamma(\Phi_l(\mathbf{x})) - \gamma(\Phi'_l(\mathbf{x}))| < (2n+1)\eta.$$

Finalmente

$$\begin{aligned}\|f(\cdot) - \Gamma_\gamma(t) \star \Theta'(t) \star (\cdot)\|_\infty &\leq \|f(\cdot) - \Gamma_\gamma(t) \star \Theta(t) \star (\cdot)\|_\infty + \|\Gamma_\gamma(t) \star \Theta(t) \star (\cdot) - \Gamma_\gamma(t) \star \Theta'(t) \star (\cdot)\|_\infty \\ &< \|f(\cdot) - \Gamma_\gamma(t) \star \Theta(t) \star (\cdot)\|_\infty + (2n+1)\eta = \frac{2n+2}{2n+3}\|f\|_\infty.\end{aligned}$$

Por tanto $\Theta'(t)$ es $\frac{2n+2}{2n+3}$-preKolmogorov para f, como queríamos demostrar. □

1.2.3. Teoremas sobre las aproximaciones

Mediante los Teoremas 1.19 y 1.20 podemos deducir el siguiente resultado.

Proposición 1.21. *Sean $f : I^n \to \mathbb{R}$ y P_f el conjunto de matrices de dimensiones $(2n+1) \times n$ que son $\frac{2n+2}{2n+3}$-preKolmogorov para f y de Kolmogorov. Entonces P_f es un conjunto **abierto** y **denso** en el conjunto de las matrices de dimensiones $(2n+1) \times n$ de Kolmogorov, utilizando la norma $\|\cdot\|_\infty$.*

Teorema 1.22 (Teorema de Aproximación Numerable de Kolmogorov)**.** *Existe una matriz $\Theta(t)$ de dimensiones $(2n+1) \times n$ y de Kolmogorov que es $\frac{2n+2}{2n+3}$-preKolmogorov para todo polinomio $p : I^n \to \mathbb{R}$ con coeficientes racionales.*

Demostración. Sea $P = \{p(\mathbf{x}) \colon I^n \to \mathbb{R} \mid p(\mathbf{x}) \in \mathbb{Q}[\mathbf{x}]\}$, que es un conjunto numerable. Consideremos pues P_q, con $q \in \mathbb{P}$, el conjunto de las matrices $\Theta(t)$ de Kolmogorov que son $\frac{2n+2}{2n+3}$-preKolmogorov para q. Sea $Q = \bigcap_{q\in\mathbb{P}} P_q$. Por la Proposición 1.21 sabemos que Q es la intersección numerable de conjuntos abiertos y densos dentro del espacio de las matrices de dimensiones $(2n+1) \times n$ de Kolmogorov, que es claramente no vacío. Gracias al Lema A.8 además sabemos que es completo. Por el Teorema de Baire A.7 el conjunto Q es denso, por ende no vacío. Basta con tomar cualquier matriz en Q. □

Teorema 1.23 (Teorema de Aproximación de Kolmogorov)**.** *Existe una matriz $\Theta(t)$ de dimensiones $(2n+1) \times n$ de Kolmogorov que es $\frac{2n+3}{2n+4}$-preKolmogorov para toda función continua $f : I^n \to \mathbb{R}$.*

Demostración. Sea $f : I^n \to \mathbb{R}$ una función continua cualquiera. Consideramos una matriz $\Theta(t)$ obtenida en el Teorema de Aproximación Numerable 1.22. Si $f \equiv 0$, ya sabemos que entonces dicha matriz será $\frac{2n+3}{2n+4}$-preKolmogorov para f. Supongamos pues que $\|f\|_\infty > 0$. Definimos la siguiente constante: $\alpha = \frac{1}{4(n+2)(2n+3)} \in (0, \frac{1}{60}]$. Gracias al Teorema A.9 sabemos que existe un polinomio de coeficientes racionales $q(\mathbf{x})$ que cumple la siguiente condición:

$$\|q(\cdot) - (1-\alpha)f(\cdot)\|_\infty < \alpha\|f\|_\infty.$$

Notamos que

1. $\|q(\cdot)\|_\infty \leq \|q(\cdot) - (1-\alpha)f(\cdot)\|_\infty + \|(1-\alpha)f(\cdot)\|_\infty \leq \alpha\|f\|_\infty + (1-\alpha)\|f\|_\infty = \|f\|_\infty$.
2. $\|q(\cdot) - f(\cdot)\|_\infty \leq \|q(\cdot) - (1-\alpha)f(\cdot)\|_\infty + \|(1-\alpha)f(\cdot) - f(\cdot)\|_\infty < \alpha\|f\|_\infty + \alpha\|f\|_\infty = 2\alpha\|f\|_\infty$.

Por la construcción de $\Theta(t)$ existe una matriz $\Gamma_\gamma(t)$ de dimensiones $1 \times (2n+1)$ que cumple las condiciones de la Definición 1.18 para $q(\mathbf{x})$ así que

1. $\|\gamma\|_\infty \leq \|q(\cdot)\|_\infty \leq \|f\|_\infty$.
2. $\|q(\cdot) - \Gamma_\gamma(t) \star \Theta(t) \star (\cdot)\|_\infty < \frac{2n+2}{2n+3}\|q(\cdot)\|_\infty \leq \frac{2n+2}{2n+3}\|f\|_\infty$.

De lo anterior se sigue que $\Theta(t)$ cumple las condiciones para ser $\frac{2n+3}{2n+4}$-preKolmogorov para f:

1. La primera ya la hemos visto: $\|\gamma\|_\infty \leq \|q\|_\infty \leq \|f\|_\infty$.
2. La segunda se obtiene al aplicar la desigualdad triangular de nuevo:

$$\begin{aligned}\|f(\cdot) - \Gamma_\gamma(t) \star \Theta(t) \star (\cdot)\|_\infty &\leq \|f(\cdot) - q(\cdot)\|_\infty + \|q(\cdot) - \Gamma_\gamma(t) \star \Theta(t) \star (\cdot)\|_\infty \\ &< \left(2\alpha + \frac{2n+2}{2n+3}\right)\|f\|_\infty = \frac{2n+3}{2n+4}\|f\|_\infty.\end{aligned}$$

□

1.2.4. Teorema de Representación de Kolmogorov-Arnold

Pasamos a demostrar el Teorema 1.12.

Demostración. Empezamos fijando $\Theta(t)$, la matriz del Teorema 1.23. Construimos de manera iterativa una sucesión de funciones $\{\gamma_l(t)\}_{l=0}^\infty$:

1. Para $l = 0$, definimos $\gamma_0(t) \equiv 0$.
2. Si tenemos definidas $\gamma_0(t), \gamma_1(t), \ldots, \gamma_p(t)$ entonces definimos $\gamma_{p+1}(t)$ de la siguiente manera: por el Teorema 1.23 sabemos que $\Theta(t)$ es $\frac{2n+3}{2n+4}$-preKolmogorov para la función

$$g(\mathbf{x}) = f(\mathbf{x}) - \left(\sum_{i=0}^{p} \Gamma_{\gamma_i}(t) \star \Theta(t) \star \mathbf{x}\right).$$

De modo que existe una función continua $a(t)$ tal que la matriz $\Gamma_a(t)$ cumple la Definición 1.18. Asignamos entonces $\gamma_{p+1}(t) = a(t)$.

Una vez hemos definido $\gamma_p(t)$ para todo $p \in \mathbb{N}$, procedemos a hacer el siguiente razonamiento: como $\gamma_p(t)$ ha sido definida usando el Teorema 1.23 con $g(\mathbf{x}) = f(\mathbf{x}) - \left(\sum_{i=0}^{p-1} \Gamma_{\gamma_i}(t) \star \Theta(t) \star (\mathbf{x})\right)$, entonces, por la segunda propiedad de ser $\frac{2n+3}{2n+4}$-preKolmogorov, se cumple que:

$$\begin{aligned}\left\|f(\cdot) - \sum_{i=0}^{p} \Gamma_{\gamma_i}(t) \star \Theta(t) \star (\cdot)\right\|_\infty &= \Bigg\|\underbrace{\left(f(\cdot) - \sum_{i=0}^{p-1} \Gamma_{\gamma_i}(t) \star \Theta(t) \star (\cdot)\right)}_{\text{Función usada en el Teorema 1.23 para generar } \gamma_p(t).} - \Gamma_{\gamma_p}(t) \star \Theta(t) \star (\cdot)\Bigg\|_\infty \\ &< \frac{2n+3}{2n+4}\left\|f(\cdot) - \sum_{i=0}^{p-1} \Gamma_{\gamma_i}(t) \star \Theta(t) \star (\cdot)\right\|_\infty.\end{aligned}$$

Aplicando la desigualdad anterior varias veces llegamos a la siguiente desigualdad para todo $p \geq 1$.

$$\left\|f(\cdot) - \sum_{i=0}^{p} \Gamma_{\gamma_i}(t) \star \Theta(t) \star (\cdot)\right\|_\infty < \left(\frac{2n+3}{2n+4}\right)^p \|f\|_\infty. \tag{1.2}$$

De nuevo por la construcción de $\gamma_p(t)$ y usando la primera propiedad de ser $\frac{2n+3}{2n+4}$-preKolmogorov tenemos que para $p \geq 2$:

$$\|\gamma_p\|_\infty \leq \left\|f(\cdot) - \sum_{i=0}^{p-1} \Gamma_{\gamma_i}(t) \star \Theta(t) \star (\cdot)\right\|_\infty < \left(\frac{2n+3}{2n+4}\right)^{p-1} \|f\|_\infty.$$

Esto implica que la serie funcional $\sum_{i=0}^{\infty}\gamma_i(t)$ es uniformemente convergente, dado que $0 < \frac{2n+3}{2n+4} < 1$. De modo que converge a una función γ continua.

Puesto que la serie $\sum_{i=0}^{\infty}\gamma_i$ converge uniformemente, la serie de matrices $\sum_{i=0}^{\infty}\Gamma_{\gamma_i}(t)$ converge uniformemente a $\Gamma_\gamma(t)$. Por la Proposición A.10, se cumple entonces que:

$$\begin{aligned}\sum_{i=0}^{\infty}\left(\Gamma_{\gamma_i}(t)\star\Theta(t)\star(\cdot)\right) &\overset{\text{Prop. A.10}}{=} \left(\sum_{i=0}^{\infty}\Gamma_{\gamma_i}(t)\right)\star\Theta(t)\star(\cdot)\\ &= \begin{pmatrix}\sum_{i=0}^{\infty}\gamma_i(t) & \dots & \sum_{i=0}^{\infty}\gamma_i(t)\end{pmatrix}\star\Theta(t)\star(\cdot)\\ &= \begin{pmatrix}\gamma(t) & \dots & \gamma(t)\end{pmatrix}\star\Theta(t)\star(\cdot) \overset{\text{Def. 1.11}}{=} \Gamma_\gamma(t)\star\Theta(t)\star(\cdot).\end{aligned}$$

Por último, utilizando la desigualdad (1.2) y la convergencia uniforme llegamos a que

$$\|f(\cdot)-\Gamma_\gamma(t)\star\Theta(t)\star(\cdot)\|_\infty = \lim_{p\to\infty}\left\|f(\cdot)-\sum_{i=0}^{p}\Gamma_{\gamma_i}(t)\star\Theta(t)\star(\cdot)\right\|_\infty \le \lim_{p\to\infty}\left(\frac{2n+3}{2n+4}\right)^p\|f\|_\infty = 0.$$

De modo que hemos obtenido las matrices que el enunciado del Teorema nos pide. □

Observación 1.24. Es crucial notar que en el Teorema 1.12 la matriz $\Theta(t)$ es universal. Es decir, dada una función $f : I^n \to \mathbb{R}$ cualquiera y la matriz $\Theta(t)$ obtenida en el Teorema 1.23, existe una matriz $\Gamma_\gamma(t)$ tal que $f(\mathbf{x}) = \Gamma_\gamma(t)\star\Theta(t)\star\mathbf{x}$.

Corolario 1.25. *Toda función $f : I^n \to \mathbb{R}^m$ es representable con dos matrices $\Theta(t)$ y $\Gamma(t)$ de dimensiones $(2n+1)\times n$ y $m\times(2n+1)$ respectivamente.*

Demostración. Mediante la matriz mencionada en la Observación 1.24 podemos escoger $(\Gamma_{\gamma_i}(t))_{i=1}^m$ y construir $\Gamma(t)$ poniendo en la fila l-ésima la matriz $\Gamma_{\gamma_l}(t)$. Entonces $f(\mathbf{x}) = \Gamma(t)\star\Theta(t)\star\mathbf{x}$. □

Capítulo 2

Limitaciones del Teorema de Kolmogorov-Arnold

En este capítulo exploraremos dos cuestiones sobre el Teorema de Representación de Kolmogorov-Arnold.

1. El Teorema afirma poder representar **toda** función continua utilizando dos matrices $\Gamma(t)$ y $\Theta(t)$, de dimensiones $1 \times (2n+1)$ y $(2n+1) \times n$. ¿Es cierto el Teorema si utilizamos tan solo matrices de dimensiones $1 \times m$ y $m \times n$ con $m < 2n+1$? La respuesta a esta cuestión se halla en el **Teorema de Sternfeld**.

2. Las funciones involucradas en el Teorema de Kolmogorov-Arnold son continuas. ¿Se cumple si exigimos que sean derivables? La respuesta se halla en el **Teorema de Vitushkin**.

2.1. Sobre la optimalidad del $2n+1$

Definición 2.1 (Subconjunto Básico)**.** Sea $\mathbb{X} \subset \mathbb{R}^n$. Decimos que $\mathbb{X}$ es básico si y solo si toda función continua $f : \mathbb{X} \to \mathbb{R}$ es simplemente representable, es decir, cumple la Definición 1.1.

Toda función simplemente representable en $\mathbb{D} \subset \mathbb{R}^n$ lo es mediante la suma de a lo sumo n funciones de una variable. Esto es porque podemos agrupar las funciones que comparten variable en una sola.

Ejemplo 2.2 (Subconjunto no básico de $\mathbb{R}^n$)**.** Como ya comprobamos en el Ejemplo 1.6, $\mathbb{R}^2$ no es básico. De hecho, I^n no es básico para $n \geq 2$, dado que siempre podemos tomar como ejemplo la función $f(\mathbf{x}) = x_1 x_2$ y aplicar un procedimiento análogo.

Ejemplo 2.3 (Subconjunto básico de $\mathbb{R}^n$)**.** Consideremos el siguiente subconjunto de $\mathbb{R}^n$: $P_r = \{\mathbf{x} \in \mathbb{R}^n \mid \mathbf{x} = (x_1, \ldots, x_n) = (x_1, 0, \ldots, 0)\}$. Veamos que es básico. Sea $f : P_r \to \mathbb{R}$ una función continua. Es evidente que:

$$f(\mathbf{x}) = f(x_1, \ldots, x_n) = f(x_1, 0, \ldots, 0).$$

Sea pues $g(x) = f(x, 0, \ldots, 0)$. Lógicamente se tiene que si $A(t) = \begin{pmatrix} g(t) & 0 & \ldots & 0 \end{pmatrix}$ entonces $f(\mathbf{x}) = A(t) \star \mathbf{x}$, de modo que f es simplemente representable. Con lo cual, P_r es básico.

Para el siguiente resultado, recordamos que un **homeomorfismo** es una aplicación continua, biyectiva y con inversa continua entre espacios topológicos. Por otro lado, una **inmersión** es una aplicación que es un homeomorfismo con su imagen.

Corolario 2.4. *Existe una inmersión $\Psi : I^n \to \mathbb{R}^{2n+1}$ cuya imagen es un conjunto básico.*

Demostración. Sea $\Theta(t)$ la matriz de tipo Kolmogorov mencionada en la Observación 1.24. Consideremos las funciones $\Phi_k(\mathbf{x}) = \sum_{i=1}^{n} \kappa_i \theta_k(x_i)$. Mostraremos que la aplicación $\Psi(\mathbf{x}) = (\Phi_1(\mathbf{x}), \ldots, \Phi_{2n+1}(\mathbf{x}))$ es la inmersión que buscamos.

- **Inmersión:** la continuidad de la función se deduce de que cada una de sus coordenadas lo es, por ser suma de funciones continuas. Para ver que es inyectiva, supongamos que no lo es. Entonces existen $\mathbf{x}_1 \neq \mathbf{x}_2$ tales que $\Psi(\mathbf{x}_1) = \Psi(\mathbf{x}_2)$, de lo que se sigue que para todo k se da que $\Phi_k(\mathbf{x}_1) = \Phi_k(\mathbf{x}_2)$. Sea $q(\mathbf{x}) = \|\mathbf{x} - \mathbf{x}_2\|$, que es una función continua en I^n tal que $q(\mathbf{x}_1) \neq q(\mathbf{x}_2)$. Por la construcción de $\Theta(t)$ existe una función auxiliar $s(t)$ tal que

$$q(\mathbf{x}) = \Gamma_s(t) \star \Theta(t) \star \mathbf{x}.$$

  De modo que

$$q(\mathbf{x}_1) = \Gamma_s(t) \star \Theta(t) \star \mathbf{x}_1 = \sum_{k=1}^{2n+1} s(\Phi_k(\mathbf{x}_1)) = \sum_{k=1}^{2n+1} s(\Phi_k(\mathbf{x}_2)) = \Gamma_s(t) \star \Theta(t) \star \mathbf{x}_2 = q(\mathbf{x}_2).$$

  Esto entra en contradicción con la definición de $q(\mathbf{x})$, de modo que Ψ es inyectiva. Por último, utilizando que I^n es compacto, $\mathbb{R}^{2n+1}$ es Hausdorff y que Ψ es continua e inyectiva deducimos que es un homeomorfismo sobre su imagen (Munkres [2000]), es decir, una inmersión.

- **Imagen básica:** sea $g : \Psi(I^n) \to \mathbb{R}$ una función continua. Si vemos que g es simplemente representable hemos terminado. Sean $f = g \circ \Psi$ y $\Gamma_\gamma(t)$ la matriz del Teorema 1.12 para $\Theta(t)$ y f. Para cualquier punto $\mathbf{y} = (y_1, \ldots, y_{2n+1}) \in \Psi(I^n)$ existe, por definición, un único $\mathbf{x} = \Psi^{-1}(\mathbf{y}) \in I^n$. Por la construcción de Ψ se cumple que la componente k-ésima de $\mathbf{y}$ es $y_k = \Phi_k(\mathbf{x})$. Con lo cual

$$g(\mathbf{y}) = f(\Psi^{-1}(\mathbf{y})) = \Gamma_\gamma(t) \star \Theta(t) \star \Psi^{-1}(\mathbf{y}).$$

  Puesto que $\Theta(t) \star \Psi^{-1}(\mathbf{y})$ es, por construcción, el vector $\Psi(\Psi^{-1}(\mathbf{y})) = \mathbf{y}$, tenemos que

$$g(\mathbf{y}) = \Gamma_\gamma(t) \star \mathbf{y}.$$

  De modo que g es simplemente representable, de lo que se sigue que $\Psi(I^n)$ es básico.

□

Enunciamos el Teorema de la dimensión de Sternfeld, cuya demostración se puede encontrar en Dzhenzher [2021].

Teorema 2.5 (Teorema de la dimensión de Sternfeld)**.** *Para $n \geq 2$ no existe ningún subconjunto de $\mathbb{R}^{2n}$ que sea homeomorfo a I^n y básico.*

Ya tenemos todas las herramientas necesarias para enunciar y demostrar el resultado fundamental de esta sección.

Teorema 2.6. *Para $n \geq 2$ no existe ninguna matriz $\Theta(t)$ de dimensiones $m \times n$ con $m < 2n + 1$, tal que para toda función continua $f : I^n \to \mathbb{R}$, exista una matriz $\Gamma_\gamma(t)$ de dimensiones $1 \times m$ que cumpla $f(\mathbf{x}) = \Gamma_\gamma(t) \star \Theta(t) \star \mathbf{x}$.*

Demostración. Supongamos que sí existen dichas matrices. Utilizando un procedimiento idéntico al del Corolario 2.4 podemos encontrar una función continua $\Psi : I^n \to \mathbb{R}^m$ cuya imagen es un conjunto básico. Dado que Ψ es un homeomorfismo entonces $\Psi(I^n) \cong I^n$ con lo cual, llamando $\mathbb{A} = \Psi(I^n)$ hemos encontrado un subconjunto de $\mathbb{R}^m$ básico y homeomorfo a I^n, lo que contradice el Corolario B.1. □

2.2. Restricciones sobre las funciones de las matrices

Supongamos que queremos representar una función con el mismo número de matrices que en el Teorema 1.10. La función que queremos representar es de n variables y está en $C^r(I^n)$, mientras que las que podemos utilizar en las matrices son de k variables y están en $C^s(I^k)$. Esta situación es distinta, dado que las funciones que tenemos permitidas en las matrices son de más de una variable. Por ende el operador $\star$ no es aplicable en este contexto. Sin embargo, podemos entender que una representación en este caso es lo siguiente:

Definición 2.7 (Representación en dos capas)**.** Sea $f : I^n \to \mathbb{R}$ una función en $C^r(I^n)$. Entonces decimos que podemos representar f en dos capas mediante un conjunto de funciones de a lo sumo k variables en C^s si y solo si existen funciones externas $g_i \in C^s(I^k)$ y funciones internas $h_{i,j} \in C^s(I^k)$ tales que:

$$f(\mathbf{x}) = \sum_{i=1}^{M} g_i \left(h_{i,1}(\mathbf{x}_{S_{i,1}}), \ldots, h_{i,k}(\mathbf{x}_{S_{i,k}}) \right), \tag{2.1}$$

donde $\mathbf{x}_{S_{i,j}}$ representa un vector formado por k coordenadas (no necesariamente distintas) elegidas del vector original $\mathbf{x} = (x_1, \ldots, x_n)$ y M el número de superposiciones.

El Teorema de Vitushkin [1977] responde a si hay garantía de poder representar todas las funciones dependiendo de n, k, r, s.

Teorema 2.8 (Teorema de la dimensión de Vitushkin)**.** *Sean $n, r, k, s \in \mathbb{N}^*$. Si se cumple que:*

$$\frac{n}{r} > \frac{k}{s}$$

entonces existen funciones $f \in C^r(I^n)$ que ***no pueden ser representadas*** *en dos capas mediante superposiciones de funciones de k variables de clase C^s.*

Veamos distintos casos:

- **Caso $s = 0$:** si exigimos tan solo continuidad a las funciones ($s = 0$), no podemos aplicar el Teorema puesto que requiere todo en $\mathbb{N}^*$. Sin embargo en este caso el Teorema 1.10 nos confirma que siempre se pueden representar.
- **Caso $s = 1$:** Dado que por el Teorema 1.10 toda función continua se puede representar mediante funciones continuas de una variable, una posible pregunta es si esto ocurre en el caso derivable. En dicho caso tendríamos $r = k = s = 1$, es decir

  $$\frac{n}{1} > \frac{1}{1} \implies n > 1.$$

  Esto cubre todos los casos no triviales. Con lo cual no se puede hacer una versión del Teorema de Representación de Kolmogorov-Arnold (Teorema 1.9) con funciones derivables.
- **Caso analítico ($r = \infty$):** Un posible enfoque sería suavizar la función f. Por ejemplo, escogiendo f analítica (que es más fuerte que exigir $r = \infty$), que es un caso que Vitushkin no cubre. Sin embargo, se ha demostrado la existencia de funciones analíticas que no pueden ser representadas mediante funciones analíticas (Vitushkin [1964]).

Capítulo 3

Kolmogorov-Arnold Networks

En este capítulo explicaremos la construcción de las KANs (*Kolmogorov-Arnold Networks*), basándonos en su reciente introducción en el artículo Liu et al. [2024a], pero desarrollando una formulación propia orientada a la implementación que expondremos y utilizaremos más adelante.

3.1. De la teoría a la práctica: Motivación de las KANs

Consideremos el problema de aproximar una función desconocida f suficientemente suave a partir de un conjunto de observaciones. Disponemos del conjunto $\{(\mathbf{x}_i, \mathbf{y}_i)\}_{i\in I} \subset \mathbb{R}^n \times \mathbb{R}^m$ que cumple $\mathbf{y}_i \approx f(\mathbf{x}_i)$.

En lugar de intentar construir f desde cero, el Teorema de Representación de Kolmogorov-Arnold y el Corolario 1.25 nos proporcionan una pauta por la que empezar: existen matrices $\Theta(t)$ y $\Gamma(t)$ tales que:

$$f(\mathbf{x}) = \Gamma(t) \star \Theta(t) \star \mathbf{x}.$$

Esta perspectiva reduce el problema a encontrar las funciones de cada una de estas matrices. Una estrategia natural es parametrizar cada función mediante una familia de funciones suaves, lo que llamaremos a partir de ahora una familia funcional. De modo que debemos obtener los parámetros que mejor se ajusten a los datos.

Sin embargo, hay un problema con este enfoque: el Teorema de Representación de Kolmogorov-Arnold únicamente garantiza que las funciones de las matrices sean continuas (y el Teorema de Vitushkin 2.8 muestra que esta restricción es estricta). Por tanto, si nos restringimos al uso de dos matrices y exigimos suavidad en las mismas, resulta imposible representar todas las funciones suaves.

3.2. Arquitectura de una KAN

Definición 3.1. Una KAN es una función descrita mediante un conjunto de matrices $(A_k(t;\theta))_{k=1}^N$ compuestas con la operación $\star$:

$$K(\mathbf{x};\theta) = A_N(t;\theta) \star A_{N-1}(t;\theta) \star \cdots \star A_1(t;\theta) \star \mathbf{x}.$$

Escribimos $K(\mathbf{x};\theta)$ (en lugar de $K(\mathbf{x})$) para enfatizar que, por lo general, una KAN depende de unos parámetros que permiten su adaptación al conjunto de observaciones. Esta arquitectura puede representarse mediante un grafo dirigido, en el que las aristas están asociadas a las funciones de las matrices (también denominadas funciones de activación). Cada vez que una arista parte de un nodo, aplica su función al valor almacenado en dicho nodo. A su vez, el valor de cada nodo se obtiene como la suma de las contribuciones de todas las aristas que inciden sobre él.

Definición 3.2. Sea $K : \mathbb{R}^n \to \mathbb{R}^m$ una KAN descrita mediante las matrices $(A_k(t;\theta))_{k=1}^N$ de acuerdo con la Definición 3.1. Supongamos que cada matriz $A_k(t;\theta)$ es de dimensiones $d_k \times d_{k-1}$. Entonces llamamos **estructura** de la KAN $K(\mathbf{x};\theta)$ a la tupla $[d_0, d_1, \ldots, d_{N-1}, d_N]$.

Observación 3.3. Dada una KAN con una estructura $[d_0, d_1, \ldots, d_{N-1}, d_N]$, su dominio está incluido en $\mathbb{R}^{d_0}$ y su imagen en $\mathbb{R}^{d_N}$.

Observación 3.4. Distintas KANs pueden representar la misma función. Por ejemplo:

$$xy = \begin{pmatrix} t^2 & -t^2 \end{pmatrix} \star \begin{pmatrix} \frac{t}{2} & \frac{t}{2} \\ \frac{t}{2} & -\frac{t}{2} \end{pmatrix} \star \begin{pmatrix} x \\ y \end{pmatrix} = \begin{pmatrix} t^2 - 23 & -t^2 + 23 \end{pmatrix} \star \begin{pmatrix} \frac{t-2}{2} & \frac{t+2}{2} \\ \frac{t+2}{2} & -\frac{t+2}{2} \end{pmatrix} \star \begin{pmatrix} x \\ y \end{pmatrix}$$

Ejemplo 3.5. La primera KAN de la Observación 3.4 tiene estructura $[2, 2, 1]$ y la siguiente representación como grafo:

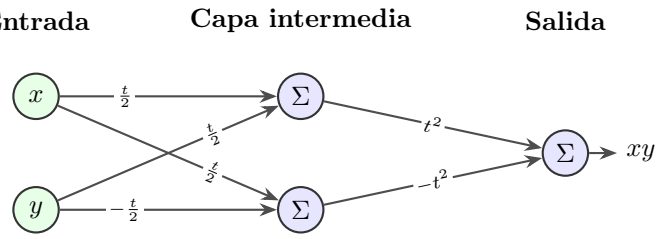

Figura 3.1: Arquitectura de la primera KAN de la Observación 3.4.

3.3. Anatomía de las aristas

3.3.1. Construcción general

Una vez definida la arquitectura global de las KANs, especificamos cómo se construyen sus aristas.

Definición 3.6. Sea $K(\mathbf{x}; \theta)$ una KAN descrita mediante las matrices $(A_k(t; \theta))_{k=1}^{N}$. Llamamos **capa** l a la matriz $A_l(t; \theta)$ y **aristas de la capa** l a las componentes de dicha matriz. Denotamos por $a_{i,j,l}(t)$ a la función de posición (i, j) en la capa l.

Configuración de las aristas

En nuestra arquitectura, cada arista tiene asociada una **función base** $b(t)$ y una combinación de hasta tres **funciones residuales**, cada una perteneciente a una familia funcional distinta. Estas últimas son las que la red debe optimizar durante el entrenamiento. Aunque se mencionen ahora dichas familias (B-splines, Polinomios de Chebyshev y Series de Fourier truncadas), se profundizará en ellas en la Sección 3.3.2.

Definición 3.7. Dados cuatro pesos w_{base}, w_{sp}, w_{ch}, w_{fo} y una función base $b(t)$, construimos una arista del siguiente modo:

$$a(t; \theta) = w_{\text{base}} b(t) + w_{\text{sp}} \text{B-spline}(t; \theta_{sp}) + w_{\text{ch}} \text{Chebyshev}(t; \theta_{ch}) + w_{\text{fo}} \text{Fourier}(t; \theta_{fo}),$$

donde $\theta = (\theta_{\text{sp}}, \theta_{\text{ch}}, \theta_{\text{fo}})$ agrupa los parámetros de cada familia funcional. Esta formulación permite desactivar cualesquiera familias utilizadas fijando en 0 los coeficientes pertinentes.

Dada una familia funcional $\mathcal{F}$, entonces $a_{i,j,l,\mathcal{F}}(t)$ es la función residual de la arista $a_{i,j,l}(t)$ (y no el peso correspondiente).
La anterior construcción es una práctica habitual en el *deep learning*: en lugar de intentar que la red aproxime directamente una función objetivo $g(t)$, hacemos que las familias residuales intenten aprender el comportamiento de la corrección $g(t) - w_{base} b(t)$. De ese modo, si se elige una función base $b(t)$ que represente una buena aproximación inicial, la parte parametrizada solo necesita modelar los detalles restantes, en lugar de tener que capturar todo el comportamiento desde cero.

Definición 3.8. Llamamos **hiperparámetros** a aquellos parámetros o configuraciones de la red que se establecen antes del entrenamiento (por ejemplo, la elección de las familias funcionales, la estructura de la red, etc). Por otro lado, llamamos **atributos** a los hiperparámetros que especifican el comportamiento de una familia funcional particular en una capa concreta (por ejemplo, el grado de los splines o el número de términos de Fourier). Estos atributos son compartidos por todas las aristas de una misma capa que utilicen dicha familia, pero pueden variar entre capas.

Observación 3.9. Es importante notar que toda la arquitectura de las KANs que desarrollaremos a lo largo del capítulo es un paradigma general y completamente extensible a otras familias de funciones. Si bien en esta formulación concretamos el uso de B-splines, polinomios de Chebyshev y series de Fourier, esto se debe única y exclusivamente a que son las familias que hemos desarrollado para nuestra implementación. No obstante, la arquitectura descrita permite integrar cualquier otra familia de funciones parametrizables aplicando exactamente los mismos principios.

3.3.2. Familias funcionales

Como hemos establecido en la sección anterior, la capacidad expresiva de una KAN reside en las familias de funciones residuales que parametrizan sus aristas. La elección de una u otra familia determina el tipo de patrones que la red puede aprender de forma eficiente. A continuación, introducimos las tres familias de funciones consideradas en nuestra arquitectura, destacando sus propiedades fundamentales.

B-splines

Los B-splines fueron la familia funcional propuesta originalmente para modelizar las aristas de las KANs. Ofrecen un equilibrio excelente entre flexibilidad y eficiencia: al estar definidos a trozos, permiten realizar modificaciones locales en la función sin alterar su comportamiento global.

Definición 3.10. Sean $(t_n)_{n=1}^K$ una sucesión no decreciente de valores reales a los que llamaremos **nudos** y $p \in \mathbb{N}^*$. Definimos las **funciones base** de los B-splines de grado p con la siguiente fórmula recursiva:

$$B_{i,0}(t) = \begin{cases} 1 & \text{si } t_i \leq t < t_{i+1} \\ 0 & \text{en caso contrario} \end{cases}$$

$$B_{i,p}(t) = \frac{t - t_i}{t_{i+p} - t_i} B_{i,p-1}(t) + \frac{t_{i+p+1} - t}{t_{i+p+1} - t_{i+1}} B_{i+1,p-1}(t),$$

donde adoptamos la convención de que $\frac{0}{0} = 0$.

Proposición 3.11. *Sean $B_{i,p}(t)$ las funciones base de la Definición anterior. Entonces:*

1. *Cada una de las funciones base es un polinomio definido a trozos. Concretamente, la base $B_{i,p}(t)$ es un polinomio de grado p con soporte en $[t_i, t_{i+p+1})$. Por ello mismo, entre dos nudos consecutivos $[t_j, t_{j+1})$, solo las funciones $B_{j-p,p}(t), \ldots, B_{j,p}(t)$ pueden ser distintas de cero.*

2. *Si un nudo t_i aparece m veces en el conjunto de nudos, entonces las funciones base con ese punto en su soporte son de clase C^{p-m} en dicho nudo.*

Los B-splines estarán compuestos por las anteriores funciones:

Definición 3.12. Dados unos nudos $(t_n)_{n=1}^K$ y un grado p, decimos que una función $r(t)$ es un B-spline si y solo si es una combinación lineal de las funciones base de los B-splines de grado p con dichos nudos. En dicho caso, si

$$r(t) = \sum_{i=1}^{n} c_i B_{i,p}(t)$$

entonces llamamos al conjunto $(c_i)_{i=1}^n$ los **puntos de control** de la función.

Observación 3.13. Si tenemos grado p y n puntos de control, entonces el número total de nudos es $n+p+1$. Esto se debe a que cada función base de grado p necesita $p+2$ nudos para definirse. Como hay n funciones base desplazadas a lo largo del vector de nudos y sus soportes se solapan, el número total de nudos necesario es $n+p+1$.

Debe quedar clara la siguiente distinción: los nudos y el grado del B-spline definen el espacio de funciones alcanzables (son atributos). Por el contrario, los puntos de control son los parámetros optimizables. Una vez fijados estos elementos, la evaluación eficiente del B-spline se realiza mediante el Algoritmo de De Boor (ver Sección C.1 en el Apéndice).

Por último, llamamos ***grid*** al conjunto de intervalos definido por los nudos.

Polinomios de Chebyshev

Los polinomios de Chebyshev surgen como una alternativa a los B-splines por tener un comportamiento global (no a trozos) en su dominio.

Definición 3.14. Definimos los polinomios de Chebyshev, $(T_n(t))_{n\in\mathbb{N}}$, con dominio $[-1,1]$, mediante la siguiente recursión

$$\begin{cases} T_0(t) = 1, \\ T_1(t) = t, \\ T_{n+1}(t) = 2tT_n(t) - T_{n-1}(t). \end{cases}$$

Fijado un grado p, para aproximar una función usamos combinaciones lineales del conjunto de funciones $(T_i(t))_{i=0}^{p}$, como en los B-splines. Los escalares que multiplican a los polinomios en dicha combinación son los coeficientes, los parámetros optimizables.

Proposición 3.15. *Los polinomios de Chebyshev $(T_n(t))_{n\in\mathbb{N}}$, cumplen las siguientes propiedades fundamentales para $t \in [-1,1]$:*

1. ***Definición trigonométrica:*** *Para cualquier $t \in [-1,1]$, se cumple que $T_n(t) = \cos(n \arccos(t))$.*
2. ***Acotación estricta:*** *Como consecuencia de la propiedad anterior, el valor del polinomio está acotado: $|T_n(t)| \leq 1$ para todo $t \in [-1,1]$. Además, sus valores máximos y mínimos locales alternan exactamente entre 1 y -1.*
3. ***Ortogonalidad****: los polinomios de Chebyshev son ortogonales con el peso $w(t) = \frac{1}{\sqrt{1-t^2}}$.*

Es importante que estén bien acotados en su dominio. Si no, la inestabilidad numérica haría imposible utilizarlos. Por otro lado, en la Observación 3.22 se detalla por qué la ortogonalidad es una ventaja.

Observación 3.16. Para valores fuera del intervalo $[-1,1]$, la definición trigonométrica se extiende mediante funciones hiperbólicas ($T_n(t) = \cosh(n \cdot \operatorname{arccosh}(t))$ para $t > 1$). Su comportamiento pues se vuelve abrupto e inmanejable, de modo que su dominio debe ser estrictamente $[-1,1]$, y esto debe manejarse para su uso en las KANs.

Series de Fourier truncadas

Pasemos a la última de las familias funcionales consideradas, las series de Fourier truncadas.

Definición 3.17 (Serie de Fourier truncada)**.** Dado un $p \in \mathbb{N}^*$ y una frecuencia fundamental $w \in \mathbb{R}_{>0}$, una función es una serie de Fourier truncada si es una combinación lineal del conjunto de funciones $\{\sin(kwt)\}_{k=1}^{p} \cup \{\cos(kwt)\}_{k=0}^{p}$.

Fijaremos $w = 2\pi$ a partir de ahora. Comentamos dos características importantes de esta familia:

- **El papel de** w: la frecuencia fundamental fija las frecuencias que puede alcanzar la serie truncada. Con $w = 2\pi$, no se puede aproximar la función $\sin(x)$ utilizando una estructura $[1,1]$ utilizando pocos términos. Sin embargo, podemos aplicar transformaciones al argumento (como se explica en el punto 3.3.3) o utilizar estructuras más complejas.

- **Periodicidad**: son la única familia periódica de las tres, lo que las hace adecuadas cuando la función objetivo posee un comportamiento periódico.

Observación 3.18. Una práctica común es considerar, en vez de la base mencionada, la base $\{\frac{\sin(kwt)}{k^2}\}_{k=1}^{p} \cup \{\frac{\cos(kwt)}{k^2}\}_{k=1}^{p} \cup \{1\}$. Esta base, que en realidad genera las mismas funciones que la anterior, introduce un término que penaliza las frecuencias altas. El objetivo de esta penalización es que el modelo sea menos propenso a utilizar esas frecuencias, simplificando su comportamiento y evitando que se adapte al ruido.

Capacidad aproximadora de las familias

Todas las familias de funciones empleadas tienen la propiedad fundamental de poder aproximar arbitrariamente bien cualquier función continua en un dominio acotado. En concreto, dada una función continua f definida sobre un intervalo cerrado $I \subset \mathbb{R}$, está garantizada la existencia de una sucesión de B-splines (de grado fijo y con una cantidad creciente de nudos) que converge uniformemente a f (de Boor [2001]). De manera análoga, si el dominio cumple que $I \subset [-1, 1]$, la función puede ser aproximada uniformemente mediante una sucesión de combinaciones lineales de polinomios de Chebyshev (Mason and Handscomb [2002]). Por último, bajo ciertas condiciones sobre el dominio, este mismo resultado de convergencia uniforme se puede alcanzar empleando polinomios trigonométricos (Stein and Shakarchi [2003]).

3.3.3. El problema del dominio

Supongamos que tenemos una KAN definida por una sucesión de matrices $(A_k(t;\theta))_{k=1}^{N}$ en las que todas las funciones pertenecen a una familia $\mathcal{F}$ de funciones cuyo dominio es $\mathbb{D}$. Al evaluar la red, la entrada que recibe la capa $l+1$ es la salida de la capa anterior l. Concretamente, si denotamos por $\mathbf{x}^{(l)}$ al vector resultante tras aplicar la matriz $A_l(t;\theta)$, el valor de su j-ésima componente viene dado por la expresión

$$x_j^{(l)} = \sum_{i=1}^{d_{l-1}} a_{j,i}^{(l)}(x_i^{(l-1)};\theta).$$

Para que el uso de la siguiente matriz $A_{l+1}(t;\theta)$ tenga sentido, es estrictamente necesario que el valor resultante de cada nodo pertenezca al dominio de las siguientes funciones, es decir, $x_j^{(l)} \in \mathbb{D}$ para todo j. Sin embargo, por la forma en la que $x_j^{(l)}$ se calcula no hay ninguna garantía de que esto ocurra.

Para solucionar este inconveniente, proponemos las dos siguientes estrategias:

- **Reescalado afín**: Mandar los *inputs* desde su intervalo actual $[a, b]$ al dominio deseado $[c, d]$ mediante la transformación afín $t \mapsto c + \frac{d-c}{b-a}(t-a)$. La principal ventaja de esto es que preserva la distribución relativa de los datos. Sin embargo, presenta un inconveniente en capas profundas ($l > 1$): el rango $[a, b]$ fluctúa durante el entrenamiento al actualizarse las capas previas, lo que obliga a recalcular los parámetros del reescalado constantemente. Si el valor obtenido tras aplicar el reescalado afín queda por debajo de c, se asigna c, y si queda por encima de d, se asigna d.

- **Reescalado global**: Aplicar una función inyectiva $T : \mathbb{R} \to [c, d]$, como tanh (adecuada, por ejemplo, para el caso Chebyshev). Es un proceso autónomo pues no requiere que los datos estén en ningún intervalo concreto ni una adaptación constante. Sin embargo, su naturaleza no lineal puede distorsionar más la distribución de los datos.

Aplicaremos estas dos técnicas en las familias de Chebyshev y Fourier. Para el reescalado global, lo hacemos del siguiente modo:

- Para el caso de Chebyshev, utilizaremos la tangente hiperbólica para que todos los *inputs* estén en $(-1, 1)$ como reescalado global. Por ser monótona conserva el orden de la distribución, y por ser suave no hace que dejen de ser derivables.

- Para las aristas de Fourier, basta con dejar el *input* por sí mismo, pues el dominio del seno y el coseno es $\mathbb{R}$.

3.4. Actualización de los atributos

Debemos contar con un mecanismo de actualización dinámica de los atributos para poder aumentar la capacidad predictiva de las KANs durante el entrenamiento. La idea del siguiente algoritmo es: primero, registrar qué datos recibe cada arista. A continuación, generar las nuevas aristas con los atributos actualizados, y por último optimizar los parámetros para que cada arista replique el comportamiento de su versión original.

Algoritmo 3.19 (Actualización de atributos)**.** *Partimos de una KAN original $K(\mathbf{x};\theta)$ en la que cada arista $a_{i,j,l}(t;\theta_{i,j,l})$ tiene asociados unos atributos $H_{i,j,l}$. Dado un conjunto de observaciones $\{(\mathbf{x}_s,\mathbf{y}_s)\}_{s\in I}\subset\mathbb{R}^n\times\mathbb{R}^m$, nuestro objetivo es construir una nueva KAN $\mathcal{K}(\mathbf{x};\tau)$, con atributos actualizados $\mathcal{H}_{i,j,l}$ para cada arista, tal que $\mathcal{K}(\mathbf{x};\tau)\approx K(\mathbf{x};\theta)$. Para encontrar los nuevos parámetros τ, procedemos de la siguiente manera:*

1. ***Evaluación global del dominio**: Evaluamos la red original $K(\mathbf{x};\theta)$ utilizando las muestras de entrada $\{\mathbf{x}_s\}_{s\in I}$. Durante este paso, registramos el conjunto de valores que circulan por cada arista, denotando por $\mathbb{D}_{i,j,l}$ al conjunto de datos de entrada que recibe la arista $a_{i,j,l}(t;\theta_{i,j,l})$.*

2. ***Actualización y creación de nuevas aristas**: Para cada arista de la red original generamos su nueva versión, $b_{i,j,l}(t;\tau_{i,j,l})$, dotada de la nueva configuración de atributos $\mathcal{H}_{i,j,l}$ (algunos atributos dependen del conjunto $\mathbb{D}_{i,j,l}$). En este instante, los parámetros $\tau_{i,j,l}$ están sin ajustar.*

3. ***Ajuste de los parámetros**: Capa por capa y para cada arista de manera independiente, ajustamos los nuevos parámetros resolviendo un problema de regresión por mínimos cuadrados sobre el conjunto de entrada $\mathbb{D}_{i,j,l}$:*

$$\tau^*_{i,j,l}=\arg\min_{\tau_{i,j,l}}\frac{1}{|\mathbb{D}_{i,j,l}|}\sum_{t\in\mathbb{D}_{i,j,l}}\big(a_{i,j,l}(t;\theta_{i,j,l})-b_{i,j,l}(t;\tau_{i,j,l})\big)^2.$$

Finalmente, asignamos a cada arista $b_{i,j,l}(t;\tau_{i,j,l})$ los parámetros $\tau^*_{i,j,l}$ y ensamblamos todas las aristas actualizadas para conformar la nueva red $\mathcal{K}(\mathbf{x};\tau)$.

3.4.1. Creación de las nuevas aristas

Empezamos explicando cómo creamos las nuevas aristas con los atributos actualizados, es decir, los pasos 1 y 2 del Algoritmo 3.19.

Actualización de la *grid* en B-splines:
Para la familia de B-splines, actualizamos la distribución de su *grid*. Dividimos la *grid* en dos partes: una ***grid* interna** formada por g intervalos (que abarca el dominio de los *inputs* de la función), y dos extensiones para que el B-spline tenga capacidad de ajuste fuera del dominio. Supongamos que tenemos una arista parametrizada por un B-spline con g intervalos internos, y deseamos reestructurar su *grid*, aumentando su resolución a g' intervalos internos (con $g'\geq g$) y adaptándola al conjunto de *inputs* actual $\mathbb{D}$. El procedimiento consta de los siguientes pasos:

1. **Delimitación del dominio:** Calculamos $a=\text{mín}(\mathbb{D})$ y $b=\text{máx}(\mathbb{D})$.

2. **Generación de mallas internas:** Construimos dos *grids* distintas en $[a,b]$ con g' intervalos:

   - Una ***grid* uniforme** ($\mathbf{grid}_{\text{unif}}$), generada mediante coordenadas estrictamente equiespaciadas.
   - Una ***grid* adaptativa** ($\mathbf{grid}_{\text{quant}}$), cuyos valores corresponden a los cuantiles del conjunto de observaciones $\mathbb{D}$.

3. **Combinación convexa:** Combinamos ambas *grids* para obtener la *grid* interna final:

$$\mathbf{grid}_{\text{conv}}=\varepsilon\cdot\mathbf{grid}_{\text{unif}}+(1-\varepsilon)\cdot\mathbf{grid}_{\text{quant}}$$

siendo $\varepsilon\in(0,1)$ un hiperparámetro.

4. **Construcción de márgenes externos:** Calculamos el tamaño medio de los nuevos intervalos de la malla interna como $h = \frac{b-a}{g'}$. Finalmente, agregamos p nudos adicionales a cada extremo de $\mathbf{grid}_{\text{conv}}$ para acomodar el grado del spline: concatenamos por la izquierda los valores $a - i \cdot h$ (con i variando desde p hasta 1) y por la derecha los valores $b + i \cdot h$ (con i variando desde 1 hasta p). Esto resulta en una *grid* de $g' + 2p + 1$ nudos, con g' intervalos internos, lo que define $g' + p$ funciones base.

Añadimos los nudos adicionales por dos motivos:

1. Queremos que la función en todo el dominio $[a, b]$ sea una combinación lineal de exactamente $p + 1$ funciones base. Si utilizáramos únicamente los nudos internos en $[a, b]$ (es decir, de t_0 a $t_{g'}$), por la Proposición 3.11 la recursión de Cox-de Boor no podría completarse en los extremos:

   - **En los primeros intervalos** (por ejemplo, $[t_0, t_1]$): Las bases que deberían ser distintas de cero y sumar en este intervalo son $B_{-p,p}, \ldots, B_{0,p}$, que lógicamente no existen. Sin embargo, para definir las bases anteriores a $B_{0,p}$ es necesario conocer nudos previos a a (es decir, $t_{-p}, \ldots, t_{-1}$).
   - **En los últimos intervalos** (por ejemplo, $[t_{g'-1}, t_{g'}]$): Las bases que deberían ser distintas de cero son $B_{g'-1-p,p}, \ldots, B_{g'-1,p}$. Para calcular la última de ellas, $B_{g'-1,p}$, la fórmula requiere nudos posteriores a b (es decir, hasta $t_{g'+p}$). Al no existir, su soporte final queda incompleto y se vuelve 0 en ese intervalo.

2. Asumir que el dominio real de la función será siempre $[a, b]$ no es realista, ya que las activaciones fluctúan en cada iteración del entrenamiento. Añadiendo las extensiones, permitimos que pueda aprender si le llegan observaciones fuera del dominio establecido. Esta extensión del dominio proporciona un margen en el que la función puede ajustarse.

Actualización de atributos en Chebyshev y Fourier:
Para estas familias, la actualización de los atributos consiste en añadir los nuevos coeficientes a la representación, inicializándolos a 0 en el paso de creación de las nuevas aristas. En el paso descrito a continuación (Subsección 3.4.2) determinamos sus valores óptimos resolviendo un problema de mínimos cuadrados.

3.4.2. Ajuste de los parámetros

Una vez hemos actualizado los atributos, necesitamos ajustar los parámetros para replicar el comportamiento de la arista. Esto es equivalente a resolver un problema de mínimos cuadrados como en una regresión lineal.

Proposición 3.20 (Ajuste de parámetros por mínimos cuadrados)**.** *Sea $\mathbb{D} \subset \mathbb{R}$ un conjunto finito de observaciones, $h(t)$ una función dada y $(G_i(t))_{i=0}^{p}$ una familia de funciones base. Entonces, encontrar los parámetros $(c_i)_{i=0}^{p}$ que minimizan el error cuadrático medio (como el tercer paso del Algoritmo 3.19) es equivalente a resolver un problema de mínimos cuadrados como en una regresión lineal.*

Demostración. Buscamos una función de la forma $g(t) = \sum_{i=0}^{p} c_i G_i(t)$. Consideramos la matriz $\mathbf{G} \in \mathbb{R}^{s \times (p+1)}$ cuyas entradas son todas las funciones base evaluadas en todos los puntos de $\mathbb{D}$:

$$\mathbf{G} = \begin{pmatrix} G_0(t_1) & G_1(t_1) & \cdots & G_p(t_1) \\ G_0(t_2) & G_1(t_2) & \cdots & G_p(t_2) \\ \vdots & \vdots & \ddots & \vdots \\ G_0(t_s) & G_1(t_s) & \cdots & G_p(t_s) \end{pmatrix}.$$

Entonces, si $\mathbf{c} = (c_0, c_1, \ldots, c_p)^\top \in \mathbb{R}^{p+1}$, se tiene que

$$\mathbf{Gc} = \begin{pmatrix} g(t_1) \\ g(t_2) \\ \vdots \\ g(t_s) \end{pmatrix}.$$

Así que, llamando $\mathbf{y} = (h(t_1), \ldots, h(t_s))^\top$, tenemos que minimizar el error cuadrático medio es equivalente a minimizar la siguiente expresión:

$$\sum_{k=1}^{s} (h(t_k) - g(t_k))^2 = \|\mathbf{y} - \mathbf{Gc}\|_2^2 .$$

Lo cual es un problema de mínimos cuadrados como en una regresión lineal. □

Observación 3.21. Si no se aplica el reescalado afín, la base de la arista original está contenida en la nueva y los coeficientes añadidos valen 0, mientras que los anteriores se mantienen intactos. Cuando se emplea el reescalado afín, la base cambia y es necesario resolver el problema de mínimos cuadrados para recalcular todos los coeficientes (véase el Ejemplo C.2).

Observación 3.22. Si utilizáramos la base de monomios $(x^i)_{i=0}^p$ en la Proposición 3.20, obtendríamos un sistema muy mal condicionado. Esto significa que la obtención de soluciones y/o aproximaciones sería muy inestable, como se describe en el artículo Gautschi and Inglese [1988]. Por otro lado, el uso de los polinomios de Chebyshev reduce este efecto, incluso con una muestra aleatoria como la que utilizaremos. Este hecho se explica en el artículo Gautschi [1983].

3.5. Flujo de entrenamiento

Situémonos en la situación descrita anteriormente, en la que queremos aproximar una función desconocida mediante un conjunto de observaciones $\{(\mathbf{x}_i, \mathbf{y}_i)\}_{i \in I} \subset \mathbb{R}^n \times \mathbb{R}^m$. Necesitamos un modo de cuantificar el nivel de acierto de nuestra KAN.

Definición 3.23. Una **función de pérdida** es una función $l : \mathbb{R}^m \times \mathbb{R}^m \to \mathbb{R}$ que recibe una tupla $(\hat{\mathbf{y}}, \mathbf{y})$ y devuelve una medida de cuán buena es la predicción $\hat{\mathbf{y}}$ con respecto al valor real $\mathbf{y}$.

Intuitivamente, la función de pérdida mide la discrepancia entre la predicción de la red y la realidad. Suponemos que la función de pérdida se anula únicamente cuando la predicción coincide exactamente con el valor real, y que valores más altos indican predicciones peores.

Definición 3.24. En el escenario de la definición anterior, definimos la **función de pérdida global** $L(\theta)$:

$$L(\theta) = \frac{1}{|I|} \sum_{i \in I} l(K(\mathbf{x}_i; \theta), \mathbf{y}_i)$$

Con una configuración fijada, nuestro objetivo es encontrar el conjunto de parámetros θ^* que minimice la función de pérdida global. Nos encontramos, pues, ante un problema de **optimización**.

Observación 3.25. Si todas las aristas de la KAN son funciones de clase C^1 con respecto a sus parámetros y a su argumento, entonces, para cada $\mathbf{x}$ fijo, la aplicación $\theta \mapsto K(\mathbf{x}; \theta)$ es de clase C^1 (por ser suma y composición de funciones de clase C^1). Si además la función de pérdida l es de clase C^1, entonces la pérdida global $L(\theta)$ también lo es.

El algoritmo de entrenamiento

El entrenamiento de una KAN consiste en un proceso iterativo: en todas las iteraciones actualizamos los parámetros, y solo en algunas de ellas los atributos:

1. ***Setup***: Partimos de un conjunto de observaciones $\{(\mathbf{x}_i, \mathbf{y}_i)\}_{i \in I} \subset \mathbb{R}^n \times \mathbb{R}^m$.

2. **Propagación hacia adelante:** Evaluamos la función de pérdida global sobre el conjunto completo de observaciones (lo que incluye el cálculo de todas las predicciones de la red).

3. **Optimización de parámetros:** Actualizamos los parámetros. Esto requiere poder encontrar el gradiente de la función de pérdida global y usar algún algoritmo concreto para actualizar los parámetros. Originalmente se propuso L-BFGS, pero existen alternativas. Explicamos más sobre este paso en el Capítulo 4.

4. **Actualización dinámica de atributos:** Cada cierto número predefinido de iteraciones, aplicamos el Algoritmo 3.19.

Este ciclo se repite continuamente, recorriendo todo el conjunto de datos durante múltiples iteraciones, que de ahora en adelante llamaremos *epochs*. El proceso se detiene cuando la función de pérdida alcanza un valor objetivo predefinido o cuando se completa el número máximo de épocas establecido.

3.6. Aproximadores universales

3.6.1. Equivalencia con las redes neuronales

Cabe preguntarse qué capacidad aproximadora tienen las KANs. ¿Son suficientemente versátiles como para aproximar cualquier función continua? Sabemos que, dada una función no polinomial σ, una función continua puede ser aproximada arbitrariamente bien mediante redes neuronales cuyas funciones de activación son todas σ (Leshno et al. [1993]). Veamos que las KAN también pueden aproximar arbitrariamente bien cualquier función continua: en una red neuronal tradicional, los parámetros optimizables son escalares que se multiplican linealmente y se suman antes de aplicar una función de activación. Es decir, una capa con entrada $\mathbf{x} \in \mathbb{R}^n$ y salida $\mathbf{y} \in \mathbb{R}^m$ calcula $y_j = \sigma\left(\sum_{i=1}^n W_{ji}x_i + b_j\right)$. Podemos pues construir explícitamente una KAN equivalente modelando cada una de sus capas mediante la composición de dos capas KAN. Para la primera capa de la KAN, definimos las funciones de las aristas como las siguientes transformaciones afines:

$$\phi_{j,i}^{(1)}(x_i) = W_{ji}x_i + \frac{b_j}{n}.$$

Al sumar estas contribuciones en los nodos de la primera capa, obtenemos exactamente el argumento previo a la función de activación, es decir,

$$z_j = \sum_{i=1}^n \phi_{j,i}^{(1)}(x_i) = \sum_{i=1}^n W_{ji}x_i + b_j.$$

Para la segunda capa de la KAN, definimos las aristas de manera que apliquen la función de activación directamente si conectan nodos correspondientes, y sean nulas en caso contrario:

$$\phi_{k,j}^{(2)}(z_j) = \begin{cases} \sigma(z_j) & \text{si } k = j, \\ 0 & \text{si } k \neq j \end{cases}$$

De esta manera, la salida final es $y_k = \sum_{j=1}^m \phi_{k,j}^{(2)}(z_j) = \sigma(z_k)$, por lo que cada capa de la red neuronal equivale a una KAN de dos capas. Como consecuencia directa del Teorema de Aproximación Universal, se deduce que las KANs también pueden aproximar cualquier función continua con precisión arbitraria.

Observación 3.26 (Extensiones teóricas). Sin embargo, el poder de aproximación de las KANs va más allá de esta herencia: su construcción permite relajar las condiciones necesarias. Esta mayor flexibilidad ha sido analizada teóricamente en el artículo Ismailov [2026], en el que se demuestran condiciones necesarias y suficientes para universalidad en el caso de las KANs.

3.7. Regularización

A priori, si queremos entrenar una KAN con un conjunto de datos, no sabemos qué estructura usar ni qué familias poner en cada arista. Por ello, usamos dos procedimientos distintos, uno para cada objetivo.

3.7.1. Poda de la red

La función $f(x,y) = xy$ se puede representar mediante una KAN de estructura $[2,2,1]$. Sin embargo, puede ser representada con menos aristas si $x, y > 0$, dado que

$$xy = \begin{pmatrix} e^t \end{pmatrix} \star \begin{pmatrix} \ln(t) & \ln(t) \end{pmatrix} \star \begin{pmatrix} x \\ y \end{pmatrix}.$$

La idea es intentar que el entrenamiento converja a representaciones con capas menos anchas.

Definición 3.27. En el contexto de la Definición 3.1, el **nodo i de la capa l** ($1 \leq l \leq N-1$) está asociado a la i-ésima fila de la matriz $A_l(t;\theta)$ (que agrupa las aristas que llegan a él desde la capa $l-1$) y a la i-ésima columna de la matriz $A_{l+1}(t;\theta)$ (que agrupa las aristas que salen de él hacia la capa $l+1$).

Nuestro objetivo es eliminar los nodos innecesarios. Para eso, necesitamos algún mecanismo que empuje a la red a concentrar sus activaciones y definir un algoritmo para podar nodos.

Definición 3.28. Sean $K(\mathbf{x};\theta)$ una KAN en el contexto de la Definición 3.2 y $\mathbb{D}_{i,j,l}$ el conjunto de *inputs* de la arista $a_{i,j,l}(t)$. Entonces definimos la **magnitud** de dicha arista como

$$|a_{i,j,l}(t)|_1 = \frac{1}{|\mathbb{D}_{i,j,l}|} \sum_{\widetilde{t} \in \mathbb{D}_{i,j,l}} |a_{i,j,l}(\widetilde{t})|.$$

A partir de la magnitud de las aristas, definimos las **importancias** de entrada y salida de cada nodo i de la capa intermedia l ($1 \leq l \leq N-1$) como la máxima magnitud de las aristas que inciden en él. Concretamente, $I_{l,i}$ es la máxima magnitud de las aristas que entran al nodo i desde la capa $l-1$, y $O_{l,i}$ la de las que salen hacia la capa $l+1$:

$$I_{l,i} = \max_{1 \leq j \leq d_{l-1}} |a_{i,j,l}|_1, \qquad O_{l,i} = \max_{1 \leq k \leq d_{l+1}} |a_{k,i,l+1}|_1.$$

Definimos entonces la **puntuación** como $P_{l,i} = \min(I_{l,i}, O_{l,i})$. Por último, definimos la **entropía de la capa** l como la entropía de la distribución que la *softmax* induce sobre las puntuaciones de sus nodos:

$$S(P_{l,\cdot}) = -\sum_{i=1}^{d_l} \rho_{l,i} \ln(\rho_{l,i}), \qquad \text{donde} \quad \rho_{l,i} = \frac{e^{P_{l,i}}}{\sum_{i'=1}^{d_l} e^{P_{l,i'}}}.$$

Gracias a esto, podemos definir una nueva función de pérdida global.

Definición 3.29. Dados los hiperparámetros $\lambda, \mu_1, \mu_2 \geq 0$, una KAN $K(\mathbf{x};\theta)$ y una función de pérdida global $L(\theta)$, la **función de pérdida global con poda** tiene la siguiente forma

$$L_{\text{poda}}(\theta) = L(\theta) + \lambda \left(\mu_1 \sum_{l=1}^{N-1} \sum_{i=1}^{d_l} P_{l,i} + \mu_2 \sum_{l=1}^{N-1} S(P_{l,\cdot}) \right).$$

Observación 3.30. La entropía $S(P_{l,\cdot})$ capta la idea de concentrar las puntuaciones en unos pocos nodos. Es máxima cuando las puntuaciones valen todas lo mismo y se reduce a medida que una domina sobre el resto.

Veamos ahora el criterio para podar.

Algoritmo 3.31 (Poda de una KAN)**.** *Sean $K(\mathbf{x};\theta)$ una KAN como en la Definición 3.2 y $P_{l,i}$ las puntuaciones de nodo de la Definición 3.28. Para podar la KAN, aplicamos los siguientes pasos:*

1. ***Puntuación relativa:*** *Para cada nodo i de la capa intermedia l ($1 \leq l \leq N-1$), calculamos su puntuación relativa a la capa, $P_{l,i}/\max_j(P_{l,j})$.*

2. ***Decisión de poda:*** *Fijado un umbral μ (por defecto $\mu = 0{,}3$), eliminamos el nodo i de la capa l si y solo si su puntuación relativa es inferior o igual al umbral, es decir*

$$\frac{P_{l,i}}{\max_j(P_{l,j})} \leq \mu.$$

Observación 3.32. En la formulación original de las KANs (Liu et al. [2024a]), el mecanismo de poda presenta dos grandes diferencias:

1. La función de pérdida se construye de manera diferente. Utilizan la suma de las magnitudes de todas las aristas y la suma de las entropías de todas las aristas. Su cálculo de la entropía de una arista coincide con el nuestro para los nodos, pero sin aplicar la misma transformación.

2. Para podar, el criterio original elige un umbral absoluto que se aplica de igual manera a todos los nodos.

Por último, una vez podada la red, es importante **refinarla**, es decir, continuar su entrenamiento con la nueva arquitectura. Dado que hay nodos y aristas que han dejado de contribuir, el modelo debe readaptar los parámetros restantes a esta nueva topología más compacta para recuperar o incluso mejorar su rendimiento.

3.7.2. Selección de familias

Presentamos la **selección de familias**, un método cuyo objetivo es poder seleccionar qué familia es mejor para cada arista, y por consiguiente poder deducir qué comportamiento tiene cada función de la representación. Del mismo modo que con la poda, empezamos definiendo nuevas funciones de las aristas.

Definición 3.33. En el contexto de la Definición 3.28, definimos, dada una arista $a_{i,j,l}(t)$, su magnitud respecto a la familia funcional $\mathcal{F}$ como

$$|a_{i,j,l,\mathcal{F}}(t)|_1^* = \frac{1}{|\mathbb{D}_{i,j,l}|} \sum_{t \in \mathbb{D}_{i,j,l}} |a_{i,j,l,\mathcal{F}}(t)|.$$

Definimos entonces la **magnitud funcional** de dicha arista, que escribimos como $|a_{i,j,l}(t)|_1^*$, como la suma de las magnitudes respecto sus funciones residuales. Combinando estos dos conceptos, definimos su **entropía funcional** como

$$S^*(a_{i,j,l}(t)) = -\sum_{\mathcal{F} \in \text{Familias}} \frac{|a_{i,j,l,\mathcal{F}}(t)|_1^*}{|a_{i,j,l}(t)|_1^*} \ln\left(\frac{|a_{i,j,l,\mathcal{F}}(t)|_1^*}{|a_{i,j,l}(t)|_1^*}\right).$$

Extendemos la definición de magnitud funcional y entropía funcional a cada capa y a cada red sumándolas por cada arista.

Definición 3.34. Dados los hiperparámetros $\lambda, \mu_1, \mu_2 \geq 0$, una KAN $K(\mathbf{x};\theta)$ y una función de pérdida global $L(\theta)$, la **función de pérdida global con selección de familias** tiene la siguiente forma

$$L_{\text{selección}}(\theta) = L(\theta) + \lambda(\mu_1 |K(\mathbf{x};\theta)|_1^* + \mu_2 S^*(K(\mathbf{x};\theta))).$$

Observación 3.35. La entropía de la red se anula si y solo si cada arista tiene activa una única familia funcional.

Para seleccionar familias, el algoritmo es más sencillo que con las podas. Si queremos quedarnos con r familias, seleccionamos de manera independiente y arista por arista las r con mayor magnitud. Una vez seleccionadas, podemos refinar la red (lo que puede aprovechar los pesos entrenados) o entrenar una red con la configuración de familias seleccionada desde 0.

Observación 3.36. La idea de combinar múltiples familias funcionales en una misma arista tiene antecedentes en la literatura. En el artículo Poole et al. [2025] se proponen las P-KANs, un marco de entrenamiento que guía el descubrimiento de las funciones de arista proyectándolas sobre distintos espacios funcionales. Aunque el objetivo de identificar la representación funcional más adecuada por arista es análogo al nuestro, el mecanismo difiere: las P-KANs parten de B-splines y analizan las funciones ya aprendidas, mientras que nuestra arquitectura entrena simultáneamente las tres familias con pesos independientes sin imponer ninguna de ellas a priori.

Capítulo 4

Implementación

En este capítulo explicaremos las partes más importantes de la implementación del marco teórico expuesto en el Capítulo 3. Lo haremos de manera simplificada, recalcando únicamente los aspectos más interesantes. Este capítulo no pretende ser un manual de cómo usar el repositorio que contiene el código. El código fuente y las instrucciones detalladas de uso pueden consultarse en el siguiente enlace:

https://github.com/trancite/tfg-kolmogorov-arnold-networks

4.1. Referencias anteriores

Existen varias implementaciones de las KANs, cada una con sus particularidades de diseño específicas. La implementación original, presentada en la documentación Liu et al. [2024b], emplea B-splines y se centra significativamente en la regresión simbólica. A raíz de dicho trabajo surgieron variaciones orientadas a resolver problemas de física, a la optimización computacional o a la integración en nuevos entornos de programación.

Un ejemplo de esto último se encuentra en el repositorio Rigas and Papachristou [2025], en el que se implementan las KANs en **JAX**. Nosotros también utilizaremos JAX (cuyos detalles se describen en la Subsección 4.2.2), introduciendo las siguientes funcionalidades:

1. **Capas heterogéneas**: a diferencia de las implementaciones convencionales, cada capa puede albergar de forma simultánea cualesquiera de las tres familias funcionales en sus aristas, permitiendo que cada arista tenga una configuración independiente dentro de la misma capa.
2. **Poda**: se utiliza en base a la Sección 3.7.1, que modifica tanto el criterio de poda como la función de pérdida respecto a la formulación original.
3. **Selección de familias**: se implementa el algoritmo descrito en la Subsección 3.7.2 para automatizar la elección de la familia funcional para cada arista. Si bien existen aproximaciones previas con objetivos similares, nuestro enfoque difiere en el modo de abordarlo.
4. **Reescalado afín**: se introduce la transformación descrita en el punto 3.3.3 para las familias funcionales Chebyshev y de Fourier.

4.2. Tecnologías y paradigma de diseño

4.2.1. Algoritmos de minimización

Tal como mostramos en la Sección 3.5, el ajuste de los parámetros en una KAN se plantea como un problema de optimización cuyo objetivo es minimizar la función de pérdida global $L(\theta)$. Calcular el mínimo analíticamente resulta inviable por la complejidad de la función, y por eso empleamos algoritmos de optimización iterativos. Nosotros utilizaremos **Adam**, por ser más rápido y menos costoso computacionalmente que el originalmente propuesto en las KANs, L-BFGS. No ahondaremos en cómo funcionan, pero sí en qué necesitan para poder utilizarse de manera práctica:

- **Evaluación**: capacidad de evaluar la función de pérdida en una cantidad masiva de datos de manera rápida y eficiente.
- **Diferenciación**: poder obtener el gradiente exacto de dicha función respecto a todos sus parámetros, de nuevo en una cantidad masiva de datos y de un modo veloz.

Para suplir estas necesidades, no nos basta con un lenguaje de programación estándar como *Python*. Sin embargo, sí podemos hacer uso de bibliotecas especializadas, capaces de operar velozmente y de derivar funciones.

4.2.2. JAX, Equinox y Optax

JAX, desarrollado por *Google*, surge como una alternativa a otras bibliotecas como TensorFlow o PyTorch. Aunque comparte el objetivo de facilitar la computación numérica eficiente y la diferenciación automática, JAX presenta una filosofía de diseño subyacente distinta. Sus características más notables incluyen:

- **Vectorización nativa**: JAX proporciona herramientas directas (como la transformación `vmap`) para vectorizar funciones. En este contexto, vectorizar significa aplicar operaciones sobre múltiples datos de forma simultánea y paralela. Si bien PyTorch cuenta con funcionalidades equivalentes, estas fueron introducidas a posteriori. Sin embargo, en JAX, esta capacidad es un pilar fundamental desde su concepción.
- **Paradigma funcional**: JAX exige que las transformaciones se apliquen sobre funciones puras, es decir, sin estado mutable ni efectos secundarios. Esto permite obtener derivadas directamente sin necesidad de construir grafos computacionales implícitos ni almacenar información oculta sobre los gradientes. Esta forma de operar permite definir funciones con la operación $\star$ y derivarlas directamente.

Acudimos a dos bibliotecas adicionales para complementar la implementación:

- **Equinox**: permite integrar la programación orientada a objetos (clases y atributos) dentro de JAX. Gracias a Equinox, podemos definir clases como `Capa` y ensamblar varias dentro de otra clase `KAN`, simplificando el código.
- **Optax**: es la biblioteca de optimización estándar en el ecosistema JAX. Se emplea para aplicar los algoritmos mencionados (Adam) de forma automatizada, encargándose de actualizar los parámetros del modelo a partir de los gradientes calculados en cada iteración.

4.3. Tensorización de las evaluaciones

Para esta explicación, llamaremos tensor a cualquier estructura de datos multidimensional nativa de JAX. En el caso de tener dos dimensiones, lo llamaremos matriz. Asimismo, llamamos eje a cada una de las dimensiones del tensor (por ejemplo, el primer eje de una matriz es el que indexa su fila). Dado un tensor T, para indicar el subtensor resultante al fijar el eje i-ésimo en la posición n_i, escribimos $T[\cdot, \ldots, \cdot, n_i, \cdot, \ldots, \cdot]$. En esta notación el símbolo $\cdot$ indica que el eje correspondiente está libre, abarcando todos sus elementos. Por último, la operación $\odot$ corresponderá a la multiplicación elemento a elemento entre dos tensores.

4.3.1. Motivación

Un posible enfoque para programar una KAN en *Python* sería construirla mediante listas anidadas: una lista de matrices, formadas a su vez por listas de filas, que serían listas de funciones individuales. Sin embargo, aunque estas funciones fuesen nativas de JAX, este diseño resultaría tremendamente ineficiente. Operar de este modo requeriría que *Python* gestionara los cálculos de forma secuencial, función a función, lo que impediría al compilador de JAX (Bradbury et al. [2018]) agrupar, fusionar y acelerar las operaciones. Intuitivamente, buscamos realizar grandes operaciones sobre grandes conjuntos de números, en lugar de ejecutar miles de pequeñas operaciones individuales.

Ejemplo 4.1. Supongamos que queremos evaluar una capa $A_l(t;\theta)$ de dimensiones 3×3, cuyas aristas son combinaciones lineales de los dos primeros polinomios de Chebyshev. En vez de almacenar una matriz de objetos de una hipotética clase `Arista Chebyshev`, resulta más eficiente considerar el tensor T_{ch} de dimensiones $3 \times 3 \times 2$, en el que el subtensor $T_{\text{ch}}[i, j, \cdot]$ es un vector con los coeficientes de la arista de posición (i, j).

4.3.2. Estructura

La implementación no es tan directa como la expuesta en el Ejemplo 4.1, dado que cada arista puede tener distintas familias funcionales activas. Se suma a esto que no sería posible asignar a cada arista una cantidad distinta de parámetros, pues todas deben tener la misma para ser un tensor nativo de JAX.
Por todas estas razones, necesitamos varios tensores distintos para almacenar la información y operar de manera eficiente. A continuación detallaremos los objetos más importantes que tiene cada capa, y cómo se ensamblan en una KAN.

Máscara funcional

Para empezar, dada una capa $A_l(t;\theta)$ de dimensiones $d_l \times d_{l-1}$, usamos lo que denominaremos una **máscara funcional**, que consiste en un tensor M de dimensiones $d_l \times d_{l-1} \times 3$. Cada familia funcional tiene asociado un índice $k \in \{0, 1, 2\}$, con el que obtenemos una matriz de dimensiones $d_l \times d_{l-1}$ al considerar $M[\cdot, \cdot, k]$. Esta matriz indica qué aristas tienen activa esa familia funcional y cuáles no: en aquellas en las que está activa, el valor es 1, y en el resto es 0. Es fundamental que la máscara funcional no sea un parámetro optimizable en ningún punto del entrenamiento.

Tensores por familia

Con la máscara funcional configurada, separamos los parámetros de cada familia funcional en tensores distintos.

- **B-spline**: si utilizamos grado p y g intervalos interiores, necesitamos dos tensores: uno que almacene los puntos de control, de dimensiones $d_l \times d_{l-1} \times (g + p)$ y otro para los nudos, de dimensiones $d_l \times d_{l-1} \times (g + 2p + 1)$.
- **Chebyshev**: con grado p únicamente necesitamos el tensor de los coeficientes, de dimensiones $d_l \times d_{l-1} \times (p + 1)$. Aparte, podemos almacenar dos tensores de dimensiones $d_l \times d_{l-1}$ si queremos utilizar el reescalado afín descrito en la Subsección 3.3.3.
- **Fourier**: si usamos p frecuencias, necesitamos un tensor de dimensiones $d_l \times d_{l-1} \times (2p+1)$ (es decir, un coeficiente para el caso $k = 0$ y, a continuación, el conjunto de frecuencias $\{1, \ldots, p\}$ con sus respectivos senos y cosenos). Lo mismo ocurre con el caso del reescalado afín, que requeriría dos matrices adicionales de $d_l \times d_{l-1}$.

Estos tensores son los responsables de almacenar la información necesaria para la evaluación. Para efectuar los cálculos de manera tensorizada (como explicamos en la introducción de esta sección), utilizamos la función `vmap` de JAX. Esta herramienta toma una función diseñada para operar sobre datos individuales e internamente la vectoriza para que actúe de forma simultánea a lo largo de los ejes de nuestros tensores, evitando el uso de bucles ineficientes.

Ejemplo 4.2. Supongamos que tenemos un tensor de aristas de Chebyshev, T_{ch}, de dimensiones $d_1 \times d_0 \times (p+1)$. Además contamos con una columna de entrada $\mathbf{x} = (x_1, \ldots, x_{d_0})^\top$ de d_0 elementos. Consideremos una función base $g_1 : \mathbb{R} \times \mathbb{R}^{p+1} \to \mathbb{R}$. Dicha función toma un escalar $x \in \mathbb{R}$ y un vector de coeficientes $\mathbf{c} = (c_0, \ldots, c_p) \in \mathbb{R}^{p+1}$, y evalúa en x el polinomio de Chebyshev definido por la combinación lineal con dichos coeficientes.

Nuestro objetivo es construir una función que nos permita realizar la operación $\star$ entre la matriz de aristas cuyos parámetros están en T_{ch} y el vector de entrada $\mathbf{x}$.

1. Definimos $g_2 = \texttt{vmap}(g_1, (0, 0))$. Dada una matriz de coeficientes C_0 de dimensiones $d_0 \times (p+1)$ (correspondiente a una rebanada horizontal de T_{ch}), la función $g_2(\mathbf{x}, C_0)$ recorre el primer eje

de $\mathbf{x}$ y el primer eje de C_0 de forma simultánea, obteniendo pares $(x[j], C_0[j, \cdot])$, y evalúa g_1 en cada uno. El resultado es un vector columna de d_0 elementos, cuya j-ésima coordenada contiene el polinomio de Chebyshev con coeficientes $C_0[j, \cdot]$ evaluado en x_j.

2. Definimos $g_3 = \texttt{vmap}(g_2, (\texttt{None}, 0))$. Ahora $\mathbf{x}$ se mantiene fijo y el `vmap` propaga por la primera dimensión de T_{ch}, aplicando g_2 a cada fila de T_{ch}. El resultado es una matriz de dimensión $d_1 \times d_0$, que tiene en la posición (i, j) la evaluación del polinomio de Chebyshev de la arista (i, j) en x_j.

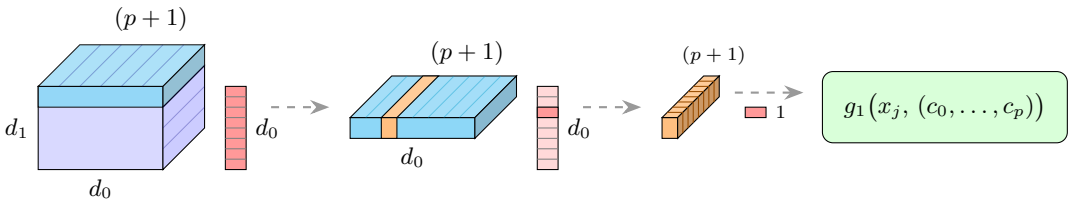

Figura 4.1: Vectorización progresiva de la evaluación de polinomios de Chebyshev mediante `vmap`. Nótese que en la segunda figura, la parte naranja del plano azul es una fila, porque `vmap` desplaza el segundo eje del tensor original al primero y el tercero al segundo.

Pesos de cada familia

Como ya sabemos por la construcción de las aristas de la Definición 3.7, cada una tiene un peso distinto y optimizable. Por ello, necesitamos un tensor más W, de dimensiones $d_l \times d_{l-1} \times 3$, en el que en la posición (i, j, k) tenemos el peso de la arista de posición (i, j) en la matriz de aristas $W[\cdot, \cdot, k]$. Usamos otro de dimensiones $d_l \times d_{l-1}$ para los pesos de la función base, que llamaremos W_b.

Evaluación de una capa

Mostramos aquí los pasos seguidos para evaluar una capa genérica basándonos en la implementación computacional ya descrita. En lugar de referirnos a cada una de las familias funcionales, utilizaremos un índice $k \in \{0, 1, 2\}$ para denotar cada una de ellas. Por tanto asumiremos que la familia k-ésima emplea p_k parámetros por arista. Partimos de los tensores previamente definidos y de un vector de observaciones de entrada $\mathbf{x} = (x_1, \ldots, x_{d_{l-1}})^\top$:

1. **Desactivación de parámetros:** para cada familia $k \in \{0, 1, 2\}$, extraemos la matriz correspondiente de la máscara funcional $M[\cdot, \cdot, k]$ (de dimensiones $d_l \times d_{l-1}$) y la expandimos a lo largo de un nuevo tercer eje, creando un tensor M_k de dimensiones $d_l \times d_{l-1} \times 1$. Multiplicamos este tensor elemento a elemento con el tensor de coeficientes T_k (de dimensiones $d_l \times d_{l-1} \times p_k$) (lo que equivale a multiplicar cada rebanada vertical de coeficientes $T_k[\cdot, \cdot, r]$ con la máscara original), obteniendo el tensor enmascarado $T'_k = T_k \odot M_k$. Dado que la máscara M no es optimizable, los coeficientes de las familias desactivadas quedan fijados a 0, evitando su contribución neta en la evaluación y, más importante, haciendo que JAX sepa que su gradiente es 0.

2. **Evaluación de familias:** utilizando el tensor enmascarado T'_k de cada familia, aplicamos nuestro procedimiento vectorizado (análogo al del Ejemplo 4.2) sobre el vector de entrada $\mathbf{x}$. Obtenemos así una matriz de evaluaciones E_k de dimensiones $d_l \times d_{l-1}$ para cada familia k.

3. **Apilamiento y combinación de aristas:** apilamos las tres matrices resultantes E_0, E_1 y E_2 a lo largo de un tercer eje para formar un tensor de evaluaciones base E, de dimensiones $d_l \times d_{l-1} \times 3$. A continuación, multiplicamos el tensor de pesos W (de dimensiones $d_l \times d_{l-1} \times 3$) por la máscara funcional original M, obteniendo $W' = W \odot M$. Ponderamos las evaluaciones calculando $E \odot W'$ y sumamos el resultado a lo largo del tercer eje (el eje de las familias). Esto resulta en una única matriz A_{res} de dimensiones $d_l \times d_{l-1}$, que representa la activación combinada de las familias para cada arista.

4. **Evaluación de la función base:** de manera independiente a las familias, evaluamos la función base genérica de la capa, aplicándola sobre cada componente del vector $\mathbf{x}$. Expandimos el vector resultante y lo multiplicamos por la matriz de pesos propios de la función base, W_b (de dimensiones $d_l \times d_{l-1}$). El resultado es una matriz B, donde cada elemento se calcula como $B_{i,j} = W_{b\,[i,j]} \cdot b(x_j)$.

5. **Agregación final:** sumamos la matriz de las familias y la de la función base para obtener la matriz de activaciones totales de la capa, $A = A_{\text{res}} + B$. Por último, sumamos los elementos de A a lo largo de su segundo eje (el correspondiente a la dimensión d_{l-1}). De este modo se agregan todos los valores, produciendo finalmente el vector de salida de la capa de dimensión $d_l \times 1$.

Evaluación de KAN

Toda KAN se estructura como una secuencia de capas. A nivel de implementación, estas capas se almacenan de forma secuencial en una lista estándar de *Python*. A diferencia de las operaciones internas de una capa, no es posible ni tiene sentido tensorizar la red completa en un único bloque masivo. Esto se debe a dos motivos fundamentales:

1. Las matrices de distintas capas poseen dimensiones diferentes, lo que imposibilita agruparlas en un tensor a primera vista.

2. La propagación de los datos es un proceso estrictamente secuencial, dado que la evaluación de la capa $l + 1$ depende por completo de la salida que acaba de calcular la capa l.

Observación 4.3 (Configuración general). Omitimos la descripción del resto de decisiones técnicas y de diseño, dado que no son de relevancia para el escrito. La mayoría de estos elementos se han implementado siguiendo fielmente la propuesta original detallada en Liu et al. [2024a].

Capítulo 5

Experimentación y resultados

En este capítulo pondremos a prueba el repositorio basado en los Capítulos 3 y 4.

5.1. Objetivo de las pruebas

Nuestros objetivos son los siguientes:

1. Analizar el rendimiento de las KANs y comparar las distintas familias funcionales.
2. Comparar los dos reescalados de la Subsección 3.3.3.
3. Comparar el rendimiento de la poda descrita en la Subsección 3.7.1 con el de la implementación original del repositorio `pykan`.
4. Poner a prueba la selección de familias (Subsección 3.7.2) y verificar si las capas heterogéneas pueden ser mejores que las homogéneas.

5.2. Configuración general

A lo largo de las pruebas la configuración será la siguiente:

- **Conjuntos de entrenamiento y evaluación**: para cada función, generaremos un conjunto de entrenamiento y uno de evaluación. Ambos tendrán la misma cantidad de muestras, serán disjuntos y estarán repartidos de manera uniforme en el dominio de la función que queremos aproximar.
- **Pérdida**: la función de pérdida general será el error cuadrático medio. Para la poda y la selección de familias, se emplearán las pérdidas específicas de las Definiciones 3.29 y 3.34.
- **Métrica**: para evaluar las redes calcularemos el **RMSE** (*Root Mean Squared Error*), cuya escala es la misma que la de la función objetivo. Para tener una mejor perspectiva calcularemos el R^2.
- **Optimizador**: como en el resto del trabajo, usaremos el optimizador Adam.

Antes de empezar, debemos explicar un inconveniente que hemos detectado en los experimentos: los peores errores de las KANs entrenadas ocurren, con mucha diferencia, en los bordes del dominio. No eliminaremos esos puntos del conjunto de evaluación, ni restringiremos la misma al interior del dominio para mitigar el efecto, pero debe tenerse presente, pues este fenómeno hace que el MAE (*Mean Absolute Error*) sea considerablemente menor que el RMSE. Por último, escribiremos 1 en el R^2 si $R^2 \geq 0{,}995$.

5.3. Experimentación

La cantidad de muestras utilizada en cada experimento variará sin seguir un patrón fijo. Sí es cierto que, a mayor dimensión del dominio, más muestras se necesitan para aprender una función. Sin embargo, creemos que es útil poder comparar las redes en contextos con más y menos datos disponibles. Todas las pruebas, a excepción de los experimentos hechos con la librería `pykan`, serán hechas 5 veces, con semillas distintas. Lo que reportaremos pues será la media de las métricas a lo largo de todas las ejecuciones.

5.3.1. Ajuste general de funciones

Mostramos la configuración general de los siguientes experimentos en la Tabla 5.1 del apéndice.

Función Objetivo	**Dominio (N)**	**Estr.**	**Familia**	**Atributos**	**RMSE (R^2)**
$e^{\sin(3x)+y^2}$	$[-1,1]^2$ (1000)	$[2,2,1]$	B-splines	$g=10$	$\mathbf{4{,}43\times10^{-3}}$ (1)
			Chebyshev	$p=12$	$1{,}06\times10^{-2}$ (1)
			Fourier	$n_{\text{freq}}=6$	$3{,}13\times10^{-2}$ (1)
$\sin(\sin(10x)+\cos(5y))$	$[-\pi,\pi]^2$ (3000)	$[2,5,3,1]$	B-splines	$g=18$	$2{,}53\times10^{-1}$ (0,84)
			Chebyshev	$p=20$	$2{,}50\times10^{-1}$ (0,85)
			Fourier	$n_{\text{freq}}=10$	$\mathbf{5{,}73\times10^{-2}}$ (0,99)
$P_6(x,y,z)$	$[-2,2]^3$ (5000)	$[3,6,4,1]$	B-splines	$g=8$	61,51 (0,99)
			Chebyshev	$p=10$	36,2 (1)
			Fourier	$n_{\text{freq}}=5$	84,39 (0,98)

Tabla 5.1: Resultados de la primera fase de los experimentos. La función del tercer ejemplo es $P_6(x,y,z)=(x+2y-z)\,(2x-y+3z+1)\,(x^2+yz-2x+4)\,(3y^2+xz-z+2)$.

Comentamos el desempeño de cada familia por separado:

- **Fourier:** Resulta ser la mejor familia para la segunda función, lo cual es lógico pues está compuesta por senos y cosenos. Cabe destacar que con una arquitectura menor, como $[2,1,1]$, obtiene un **RMSE** de $4{,}1\times10^{-1}$ (un factor $\times7{,}16$ peor), pese a que técnicamente la función objetivo sí sea representable mediante series de Fourier truncadas con esa estructura. Esto sugiere que es necesario utilizar estructuras más complejas, ya sea para encontrar representaciones más sencillas o para abarcar una cantidad mayor de frecuencias. Por otro lado, queda relegada al último puesto en la primera y tercera prueba, lo que es un indicio de que le cuesta modelar crecimientos polinómicos o exponenciales comparada con el resto de familias.

- **B-splines:** Ha sido la mejor en la primera prueba, lo cual creemos que sucede por su capacidad de ajuste local. En dicho ejemplo logra batir a Chebyshev y Fourier, estando compuesta la función objetivo por componentes trigonométricas y cuadráticas. Para la tercera función (el polinomio) ofrece un resultado intermedio. Sin embargo, en la segunda prueba, se evidencia su límite: al igual que Chebyshev, fracasa a la hora de capturar oscilaciones muy rápidas, obteniendo ambos un R^2 casi idéntico ($\approx 0{,}85$). Solucionar esto requeriría, probablemente, un número mayor de datos y una *grid* más densa, para poder captar las fuertes oscilaciones de la segunda función objetivo. En las imágenes 5.1 y 5.2 mostramos visualmente una prueba de dicho escenario en el que se ajusta de manera muy artificial a los datos.

- **Chebyshev:** Como era de esperar, demuestra ser la mejor opción para el polinomio, con bastante diferencia. En la primera función ofrece un buen rendimiento (quedando en segundo

lugar), lo que prueba que puede adaptarse de manera efectiva a comportamientos no polinómicos. No obstante, al enfrentarse a la alta frecuencia de la segunda función, sufre exactamente el mismo problema de capacidad que los B-splines, demostrando ser insuficiente para modelar oscilaciones tan concentradas.

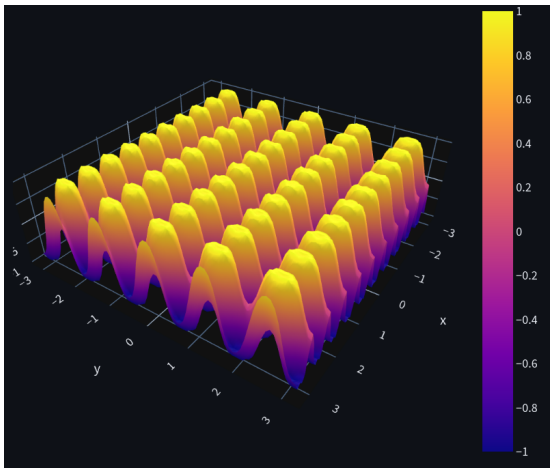

Figura 5.1: Función objetivo del segundo experimento de la Tabla 5.1.

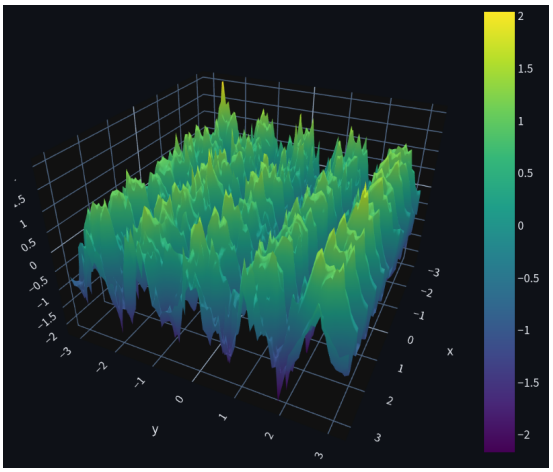

Figura 5.2: KAN entrenada con B-splines en el segundo experimento de la Tabla 5.1

En conjunto, estos primeros resultados confirman que no existe una familia funcional universalmente superior, sino que el rendimiento depende fuertemente de la naturaleza de la función objetivo: los B-splines destacan por su versatilidad y capacidad de adaptarse a distintos escenarios, Fourier es la mejor para aproximar oscilaciones muy pronunciadas y concentradas, y Chebyshev ofrece un equilibrio similar al de los B-splines, destacando más en los polinomios.

5.3.2. Comparación de reescalados

Para empezar esta comparativa, repetiremos los experimentos de la Tabla 5.1, esta vez comparando ambos reescalados.

Chebyshev Afín	Chebyshev Global	Fourier Afín	Fourier Global
$9{,}78 \times 10^{-3}$ (1)	$7{,}37 \times 10^{-2}$ (0,99)	$3{,}12 \times 10^{-2}$ (1)	$5{,}06 \times 10^{-1}$ (0,87)
$1{,}52 \times 10^{-1}$ (0,93)	$3{,}78 \times 10^{-1}$ (0,68)	$5{,}57 \times 10^{-2}$ (0,99)	$7{,}04 \times 10^{-1}$ ($-0{,}05$)
47,56 (0,99)	100,88 (0,97)	84,39 (0,98)	403,84 (0,57)

Tabla 5.2: Comparación de los dos reescalados con los ejemplos de la Tabla 5.1. Se mantiene la configuración general. Salvo en los casos que el reescalado afín coincide con el de la tabla 5.1, se han utilizado semillas distintas.

Es evidente que el reescalado afín es consistentemente superior al global, con diferencias particularmente notables en las redes de Fourier. En las imágenes 5.4 y 5.5 se puede observar cómo el reescalado global genera patrones forzados que no capturan suavemente la función objetivo. Creemos que esto obedece a razones distintas según la familia funcional. Para Chebyshev, la causa radica en que los dominios empleados son bastante pequeños respecto a los crecimientos pronunciados que presentan. En estos escenarios, el reescalado global comprime de manera mucho más agresiva la señal de entrada, dificultando que la red distinga las variaciones sutiles en el comportamiento de la función. Por el contrario, el reescalado afín se adapta dinámicamente a la dispersión real de las activaciones en cada capa, lo que permite aprender estas transiciones con mayor precisión. Dado que los valores de activación no se dispersan excesivamente en dominios acotados, el reescalado afín no comete grandes fallos, permitiendo que el aprendizaje mejore. Para

Fourier, intuimos que el problema es de naturaleza distinta. Sin un reescalado afín, la cantidad de frecuencias disponibles para cada arista se limita a múltiplos de 2π, lo que reduce la flexibilidad de las funciones disponibles. Con el reescalado afín, estas frecuencias pueden variar y adaptarse de manera idónea, según los requerimientos de la función objetivo, dando lugar a una representación mucho más precisa.

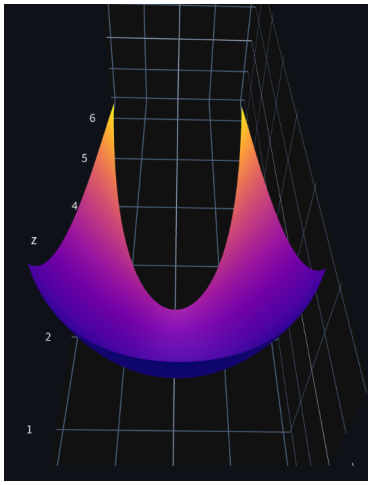

Figura 5.3: Función objetivo del primer experimento de la Tabla 5.1.

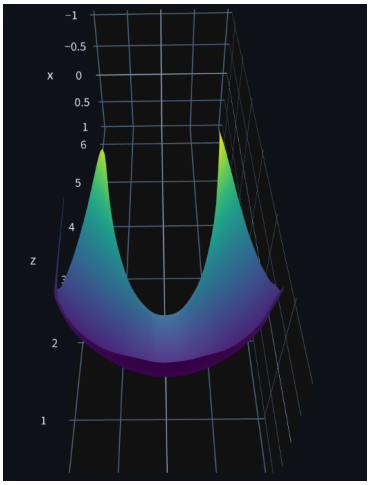

Figura 5.4: KAN entrenada con Fourier y reescalado afín para el primer experimento de la Tabla 5.1.

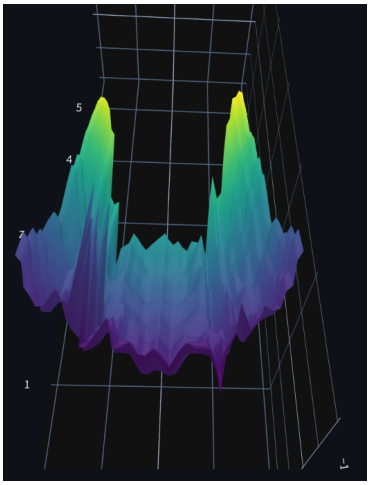

Figura 5.5: KAN entrenada con Fourier y reescalado global para el primer experimento de la Tabla 5.1.

Experimentos adicionales

Para poner a prueba las hipótesis planeadas anteriormente, hemos diseñado dos experimentos adicionales. En uno, utilizamos una función compleja, en un dominio más amplio, y cuyo crecimiento es estable. En el otro, utilizamos una función con muchas variaciones concentradas en un dominio mucho menor. Esperamos que, en el caso de Chebyshev, la diferencia entre el reescalado global y el afín se reduzca en el primero de los experimentos y se mantenga en el segundo. Para Fourier, que se mantenga en ambos.

Función Objetivo	Dominio	Estr.	Familia	Reescalado	RMSE (R^2)	ms/epoch
$\frac{2^x+2^{10-y}z}{1+xy+xz+yz}$	$[0,10]^3$ (5000)	$[3,6,3,1]$	Chebyshev	Afín	40,86 (0,75)	50,30
				Global	**37,03** (0,85)	43,83
			Fourier	Afín	**22,72** (0,94)	52,26
				Global	88,28 (0,16)	41,64
$(e^{\frac{xy}{2}}+x^y)\sin(x^2)$	$[0,3]^2$ (5000)	$[2,5,1]$	Chebyshev	Afín	$\mathbf{1{,}31\times10^{-1}}$ (1)	28,34
				Global	$6{,}30\times10^{-1}$ (1)	19,91
			Fourier	Afín	$\mathbf{2{,}13\times10^{-1}}$ (1)	29,09
				Global	1,83 (0,96)	19,57

Tabla 5.3: Comparativa de rendimiento utilizando las familias de Chebyshev y Fourier entre el reescalado afín y el reescalado global. Se destaca en negrita el mejor resultado general para cada familia y función.

Los resultados son los que anticipábamos. De hecho, la primera prueba es la única de todas las que hemos hecho hasta ahora en la que el reescalado global supera al afín. Debe notarse que lo hace, pero de manera ligera, y que teniendo en cuenta la varianza de los errores (que puede

ser consultada en el repositorio), no podemos concluir definitivamente que sea mejor el reescalado global que el afín en este caso.

Velocidad de ejecución

Por último, en lo referente al coste computacional, observamos que el uso del reescalado afín (y el hecho de aumentar el número de capas, que conlleva más reescalados) incrementa de forma consistente el tiempo de entrenamiento por *epoch*. Este aumento se debe al coste de la actualización dinámica de atributos asociada a la lectura del máximo y el mínimo en cada nueva capa. Sin embargo, dado que la diferencia observada en tiempos de ejecución es asumible y la ganancia en precisión (RMSE) es notable, todo apunta a que este proceso no constituye un inconveniente computacional y justifica plenamente su implementación.

5.3.3. Evaluación de la poda

En este apartado evaluaremos la poda de nuestra implementación y la compararemos con la de la librería `pykan`. Las pruebas consisten en los siguientes pasos:

1. Entrenar la red con la función de pérdida de poda, utilizando N iteraciones. En esta parte, actualizamos los atributos aumentando g hasta que llegue a su valor final.
2. Podar la red.
3. Refinar la red podada, mediante otras N iteraciones, con dos actualizaciones de la *grid*.

Para que la comparación entre librerías sea justa, todos los experimentos compartirán las configuraciones pertinentes. Esto no incluye, por ejemplo, al umbral de poda, que lógicamente depende de qué tipo de poda usemos. Por otro lado, `pykan` utiliza el algoritmo L-BFGS, mientras que nosotros usamos Adam. Las iteraciones de uno no son comparables con las de otro, así que las ajustaremos de manera que ambas librerías tengan el mismo tiempo disponible de entrenamiento. A continuación mostramos los resultados.

Función Objetivo	Dominio	Método	Estr.	RMSE (R^2)	s
$\exp\left(\frac{\sin(x_1^2+x_2^2)+\sin(x_3^2+x_4^2)}{2}\right)$	$[-1,1]^4$ (1000)	Original	$[4,4,4,1]$	$6{,}03 \times 10^{-3}$ (1)	-
		Poda	$[4,3,1,1]$	$\mathbf{5{,}22 \times 10^{-3}}$ (1)	58,46
		Poda `pykan`	$[4,4,1,1]$	$2{,}6 \times 10^{-1}$ (0,78)	91
xy	$[-5,5]^2$ (5000)	Original	$[2,10,1]$	$\mathbf{7{,}52 \times 10^{-3}}$ (1)	-
		Poda	$[2,3,1]$	$1{,}11 \times 10^{-2}$ (1)	117,53
		Poda `pykan`	$[2,3,1]$	7×10^{-3} (1)	102
$\sqrt{(x_1-x_2)^2+(x_3-x_4)^2}$	$[-2,2]^4$ (4000)	Original	$[4,3,3,1]$	$1{,}03 \times 10^{-2}$ (1)	-
		Poda	$[4,3,1,1]$	$9{,}75 \times 10^{-3}$ (1)	137
		Poda `pykan`	$[4,2,2,1]$	$8{,}8 \times 10^{-2}$ (1)	99

Tabla 5.4: Comparativa entre la poda propia y la poda original de `pykan`: estructura final, RMSE tras refinamiento y tiempo de cómputo.

En conjunto, bajo restricciones de tiempo comparables, nuestro método nunca produce más nodos que `pykan` y alcanza un error medio menor, logrando superar la precisión de la librería original en dos de los tres escenarios analizados. Conviene matizar que reducir nodos no implica reducir aristas en la misma medida: en el tercer experimento, por ejemplo, nuestra red conserva más conexiones que la de `pykan` pese a tener el mismo número de nodos. Esto se debe, posiblemente, a que nuestra función de pérdida penaliza la actividad de los nodos y no la de las aristas.

- El hecho de que una estructura obtenida tras podar converja bien no garantiza que, al entrenar desde cero con esa misma estructura, se obtenga el mismo resultado. De hecho, en la tercera prueba, si empezamos con una estructura $[4, 3, 1, 1]$ y entrenamos 4000 *epochs*, los resultados son peores (4×10^{-2}).
- Examinando las pruebas con nuestra implementación, creemos que una poda más agresiva puede traducirse en un error mayor, sobre todo si la red no dispone de suficientes iteraciones de refinamiento tras eliminar varios nodos de golpe. De hecho, en algunas ejecuciones con distintas semillas nuestra poda reduce la capa intermedia hasta la estructura $[2, 2, 1]$, eliminando 8 nodos. Aunque el producto sea representable fácilmente con dicha estructura, una reducción tan drástica de golpe exige más iteraciones de refinamiento para reajustar los parámetros restantes, lo que tiende a producir errores mayores.
- Debe recordarse que esta comparación no es estrictamente directa, pues como ya hemos dicho, ambos métodos emplean optimizadores distintos (Adam frente a L-BFGS). Así pues, no podemos afirmar que nuestro método sea intrínsecamente superior en cualquier escenario, sino que, bajo estas condiciones, tiende a ofrecer un mejor compromiso entre tiempo de entrenamiento y error obtenido.

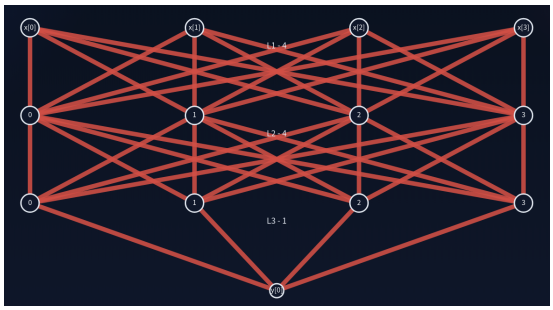

Figura 5.6: KAN antes de podar del primer experimento de la Tabla 5.4.

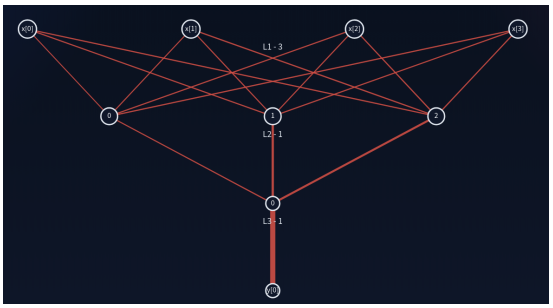

Figura 5.7: KAN podada del primer experimento de la Tabla 5.4. El grosor está escalado en función de la magnitud (3.28) de cada arista.

5.3.4. Selección de familias

Para evaluar la selección de familias, lo hacemos en dos etapas. Primero, entrenamos 2000 *epochs* usando la pérdida de selección (Definición 3.34) para definir la arquitectura óptima. Después, reentrenamos esa estructura desde cero durante otros 2000 *epochs* ya sin dicha pérdida. El refinamiento de la red, con los parámetros arrastrados de la fase de selección, parece funcionar mucho peor (a diferencia del caso de la poda), y por eso no lo consideramos.

Función Objetivo	Dominio / (Puntos)	Estr.	Configuración	Familias	Error
$\frac{x+y}{1+xy}$	$[0,\ 10]^2$ (5000)	$[2, 4, 1]$	Mejor homogénea	Chebyshev	$2{,}88 \times 10^{-2}$ (1)
			Selección	C, F	$2{,}03 \times 10^{-2}$ (1)
$5\sin(5x) + (y+1)^3 + 10\log(z+1)\cos(z^2)$	$[0,\ 5]^3$ (3000)	$[3, 1]$	Mejor homogénea	B-splines	5,08 (0,99)
			Selección	B	5,08 (0,99)
$\tanh(3(x_1^2 + x_2^2 + x_3^2 - 1))$	$[-2,\ 2]^3$ (10000)	$[3, 2, 1]$	Mejor homogénea	B-splines	$5{,}80 \times 10^{-3}$ (1)
			Selección	C	$7{,}5 \times 10^{-3}$ (1)

Tabla 5.5: Comparativa de rendimiento entre redes homogéneas y redes heterogéneas optimizadas mediante nuestro algoritmo de selección dinámica de familias. *C, F, B* denotan Chebyshev, Fourier y B-Splines respectivamente. Reportamos la selección de la primera ejecución.

Los resultados de la Tabla 5.5 muestran un panorama mixto para la selección de familias. En el segundo ejemplo, el algoritmo identifica correctamente la mejor configuración homogénea. Cabe

matizar, sin embargo, que esta no es la mejor configuración posible: hemos comprobado que una red heterogénea fijada ([Fourier, Chebyshev, B-splines]) consigue un error ligeramente inferior (4,77) al de los B-splines en solitario, lo que confirma que sí existe margen de mejora mediante heterogeneidad en este caso, pero que nuestro algoritmo de selección no ha sido capaz de encontrarlo. En el tercer ejemplo, pese a que los B-splines logran el mejor error homogéneo, la selección los descarta y se queda con Chebyshev en solitario, con un error ligeramente superior ($\times$1,29 veces peor). Sin embargo, en el primer ejemplo el resultado es más optimista: la combinación encontrada (Chebyshev y Fourier) obtiene un error menor que el de la mejor familia homogénea ($\times$1,42 mejor), aunque ambos errores sean considerablemente bajos.

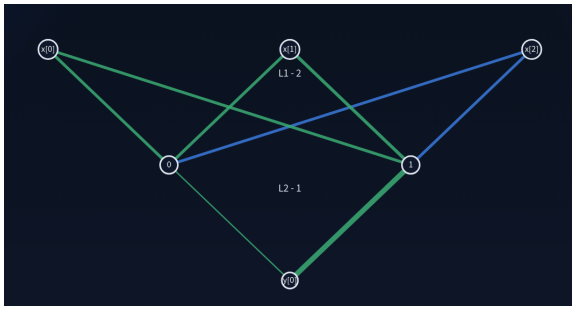

Figura 5.8: Ejemplo de una ejecución (distinta de la mostrada en la Tabla 5.5) de selección de familias en el tercer experimento de dicha tabla En verde se muestran las aristas de Chebyshev, en azul las de Fourier. El grosor está escalado en función de la magnitud (Definición 3.28) de cada arista.

Creemos que la profundidad de la red afecta significativamente a la selección de familias: en arquitecturas profundas, los B-splines casi nunca son seleccionados, mientras que Chebyshev y Fourier sí lo son. Cuantas más capas preceden a una arista, más difícil resulta que su dominio de entrada se mantenga estable: lo que recibe una arista es el valor de su nodo de origen, que a su vez es la suma de las aristas que llegan a él desde la capa anterior. Como estas últimas dependen, del mismo modo, de capas anteriores, los cambios producidos en cada actualización de parámetros se propagan y se amplifican a través de las capas. De las tres familias, los B-splines son los que más dependen de su configuración de atributos para evaluarse correctamente, ya que esta no solo delimita el dominio, sino que determina la distribución de las funciones base sobre él, dependiendo así de la distribución de los datos recibidos. Por eso son los que más sufren esta inestabilidad. Chebyshev y Fourier, en cambio, palian el problema mucho más rápido durante el entrenamiento, pues su reescalado afín solo necesita recalcular el máximo y el mínimo de sus entradas para adaptarse al nuevo dominio, frente a la reconstrucción completa del *grid* que exigen los B-splines. Por ello, en redes profundas la selección de familias tiende a descartar los B-splines en favor de Chebyshev y Fourier.

A esto se suma una segunda observación sobre el crecimiento de los atributos: aumentar la capacidad de una familia (más nudos en B-splines, más términos en Chebyshev o Fourier) no garantiza una mejora proporcional ni comparable entre familias. Es posible (y creemos que generalmente es así) que, con un mismo número de parámetros N, Chebyshev supere a los B-splines, pero que al duplicar dicha capacidad a $2N$ sean los B-splines los que pasen a ofrecer un resultado superior. Si esto es así, la convergencia de cada familia ante el aumento de su capacidad dependería fuertemente del punto de partida, lo que sugeriría que, para una comparación más justa entre familias (y para que el algoritmo de selección las seleccione correctamente), conviene hacer crecer menos los atributos.

Más allá de la precisión final, conviene señalar que la selección de familias puede aportar valor también a otro nivel: no solo el de la aproximación numérica, sino el simbólico. Conocer qué familia predomina en cada arista de una red ya entrenada y podada puede dar pistas sobre la naturaleza de la función subyacente (por ejemplo, un predominio de Fourier sugeriría patrones periódicos, mientras que un predominio de B-splines podría indicar comportamientos localizados o discontinuos). Explorar esta dimensión interpretativa, junto con una inicialización y un calendario de crecimiento de atributos más cuidadosos, queda como una línea natural de trabajo futuro.

Capítulo 6

Bibliografía

A. S. Besicovitch. On the linear independence of fractional powers of integers. *Journal of the London Mathematical Society*, 15(1):3–6, 1940. doi: 10.1112/jlms/s1-15.1.3.

J. Bradbury, R. Frostig, P. Hawkins, M. J. Johnson, C. Leary, D. Maclaurin, G. Necula, A. Paszke, J. VanderPlas, S. Wanderman-Milne, and Q. Zhang. JAX: Composable transformations of Python+NumPy programs, 2018. URL https://github.com/jax-ml/jax. Accessed: 2026-06-11.

C. de Boor. *A Practical Guide to Splines*. Springer, revised edition, 2001. doi: 10.1007/978-1-4612-6333-3.

S. Dzhenzher. A simpler proof of Sternfeld's theorem. *arXiv preprint arXiv:2110.15565*, 2021. URL https://arxiv.org/abs/2110.15565.

S. Dzhenzher and A. Skopenkov. A structured proof of Kolmogorov's superposition theorem. *arXiv preprint arXiv:2105.00408*, 2021. URL https://arxiv.org/abs/2105.00408.

W. Gautschi. The condition of Vandermonde-like matrices involving orthogonal polynomials. *Linear Algebra and its Applications*, 52:293–300, 1983. doi: 10.1016/0024-3795(83)80020-2.

W. Gautschi and G. Inglese. Lower bounds for the condition number of Vandermonde matrices. *Numerische Mathematik*, 52(3):241–250, 1988. doi: 10.1007/BF01398696.

V. Ismailov. Necessary and sufficient conditions for universality of Kolmogorov–Arnold networks. *arXiv preprint arXiv:2604.23765*, 2026. URL https://arxiv.org/abs/2604.23765.

M. Leshno, V. Y. Lin, A. Pinkus, and S. Schocken. Multilayer feedforward networks with a nonpolynomial activation function can approximate any function. *Neural Networks*, 6(6):861–867, 1993. doi: 10.1016/S0893-6080(05)80131-5.

Z. Liu, Y. Wang, S. Vaidya, F. Ruehle, J. Halverson, M. Soljačić, M. Tegmark, and I. L. Chuang. KAN: Kolmogorov–Arnold networks. *arXiv preprint arXiv:2404.19756*, 2024a. URL https://arxiv.org/abs/2404.19756.

Z. Liu, Y. Wang, S. Vaidya, F. Ruehle, J. Halverson, M. Soljačić, M. Tegmark, and I. L. Chuang. pykan documentation, 2024b. URL https://kindxiaoming.github.io/pykan/.

J. C. Mason and D. C. Handscomb. *Chebyshev Polynomials*. Chapman and Hall/CRC, 2002. doi: 10.1201/9781420036114.

J. R. Munkres. *Topology*. Prentice Hall, 2nd edition, 2000. ISBN 978-0131816299.

P. A. Ostrand. Dimension of metric spaces and Hilbert's problem 13. *Bulletin of the American Mathematical Society*, 71(4):619–622, 1965. doi: 10.1090/S0002-9904-1965-11363-5.

A. Poole, S. McArthur, and S. Kumar. Projective Kolmogorov-Arnold neural networks (P-KANs): Entropy-Driven functional space discovery for interpretable machine learning, 2025.

S. Rigas and M. Papachristou. jaxkan: A unified JAX framework for Kolmogorov–Arnold networks. *Journal of Open Source Software*, 10(108):7830, 2025. doi: 10.21105/joss.07830.

E. M. Stein and R. Shakarchi. *Fourier Analysis: An Introduction*. Princeton University Press, 2003. ISBN 978-0691113845.

A. G. Vitushkin. A proof of the existence of analytic functions of several variables not representable by linear superpositions of continuously differentiable functions of fewer variables. *Doklady Akademii Nauk SSSR*, 156(6):1258–1261, 1964. URL https://www.mathnet.ru/eng/dan29742.

A. G. Vitushkin. On representation of functions by means of superpositions and related topics. *L'Enseignement Mathématique*, 23(3-4):255–320, 1977. URL https://www.e-periodica.ch/digbib/view?pid=ens-001%3A1977%3A23%3A%3A401.

Apéndice A

Teorema de Representación de Kolmogorov-Arnold

A.1. Resultados preliminares

Proposición A.1. *Sea $\{A_k(t)\}_{k\in\mathbb{S}}$ una familia de matrices de funciones continuas de dimensión $m \times n$ y $\mathbb{S} \subset \mathbb{N}$ un conjunto finito de índices. Entonces*

$$\sum_{k\in\mathbb{S}}(A_k(t) \star \mathbf{x}) = \left(\sum_{k\in\mathbb{S}} A_k(t)\right) \star \mathbf{x}. \tag{A.1}$$

Demostración. Sea $\mathbf{x} = (x_1, \ldots, x_n) \in \mathbb{R}^n$. Por la definición del operador $\star$, el vector resultante de $A_k(t) \star \mathbf{x}$ es un elemento de $\mathbb{R}^m$ cuya componente i-ésima es $\sum_{j=1}^n a_{i,j,k}(x_j)$. Aplicando la definición de suma de vectores en $\mathbb{R}^m$, tenemos:

$$\begin{aligned}\sum_{k\in\mathbb{S}}(A_k(t) \star \mathbf{x}) &= \sum_{k\in\mathbb{S}} \begin{pmatrix} \sum_{j=1}^n a_{1,j,k}(x_j) \\ \vdots \\ \sum_{j=1}^n a_{m,j,k}(x_j) \end{pmatrix} \\ &= \begin{pmatrix} \sum_{k\in\mathbb{S}} \left(\sum_{j=1}^n a_{1,j,k}(x_j)\right) \\ \vdots \\ \sum_{k\in\mathbb{S}} \left(\sum_{j=1}^n a_{m,j,k}(x_j)\right) \end{pmatrix}.\end{aligned}$$

Dado que $\mathbb{S}$ es un conjunto finito, podemos intercambiar el orden de los sumatorios:

$$\begin{pmatrix} \sum_{j=1}^n \left(\sum_{k\in\mathbb{S}} a_{1,j,k}(x_j)\right) \\ \vdots \\ \sum_{j=1}^n \left(\sum_{k\in\mathbb{S}} a_{m,j,k}(x_j)\right) \end{pmatrix} = \begin{pmatrix} \sum_{j=1}^n \left(\sum_{k\in\mathbb{S}} a_{1,j,k}\right)(x_j) \\ \vdots \\ \sum_{j=1}^n \left(\sum_{k\in\mathbb{S}} a_{m,j,k}\right)(x_j) \end{pmatrix}.$$

Por último, dado que la matriz $S(t) = \sum_{k\in\mathbb{S}} A_k(t)$ es, por definición, la matriz cuyas entradas son las funciones continuas $s_{i,j}(t) = \sum_{k\in\mathbb{S}} a_{i,j,k}(t)$, podemos sustituir en la anterior expresión

$$\begin{pmatrix} \sum_{j=1}^n s_{1,j}(x_j) \\ \vdots \\ \sum_{j=1}^n s_{m,j}(x_j) \end{pmatrix} = S(t) \star \mathbf{x} = \left(\sum_{k\in\mathbb{S}} A_k(t)\right) \star \mathbf{x}.$$

lo que concluye la demostración. □

Demostramos a continuación varias propiedades de los intervalos de la Definición 1.14, que utilizaremos en la Proposición 1.15 y el Lema 1.16.

Proposición A.2. *Los intervalos de cualquier capa cumplen las siguientes dos condiciones:*

1. *Tienen longitud* $\frac{2n}{N}$.
2. *Son disjuntos dos a dos y dos intervalos consecutivos tienen un ancho entre ellos de* $\frac{1}{N}$.

Demostración. 1. Solo debemos restar los extremos de cada intervalo. Para $I_{j,k}(N)$,

$$\text{ancho} = \frac{(2n+1)j+k+2n}{N} - \frac{(2n+1)j+k}{N} = \frac{2n}{N}.$$

2. Debemos restar el inicio de $I_{j+1,k}(N)$ con el final de $I_{j,k}(N)$. En este caso

$$\text{hueco} = \left(\frac{(2n+1)(j+1)+k}{N}\right) - \left(\frac{(2n+1)j+k}{N} + \frac{2n}{N}\right) = \frac{1}{N} > 0.$$

Dado que la anchura del hueco es mayor que 0, no particionan I^1.

□

Proposición A.3. *En toda capa el* 0 *está contenido en el intervalo con* $j = -1$ *y el* 1 *en el intervalo con* $j = \left\lfloor \frac{N-k}{2n+1} \right\rfloor$. *De modo que todo punto de* I^1 *está en el hueco entre dos intervalos o en la capa.*

Demostración.

- **0 en el intervalo** $j = -1$: basta con notar que

$$l_{-1,k}(N) = \frac{k-(2n+1)}{N} \leq 0 \leq \frac{k-1}{N} = u_{-1,k}(N)$$

- **1 en el intervalo** $j = \left\lfloor \frac{N-k}{2n+1} \right\rfloor$: claramente tenemos que $j \leq \frac{N-k}{2n+1} < j+1$. Con lo cual

$$l_{j,k}(N) = \frac{(2n+1)j+k}{N} \leq \frac{(N-k)+k}{N} = 1 = \frac{N}{N}.$$

Por otro lado, por definición de j, tenemos que $(2n+1)(j+1) > N-k$, es decir, $(2n+1)j + 2n+1+k > N$. Así que $(2n+1)j + 2n + k \geq N$ (por ser ambos enteros). Por tanto,

$$u_{j,k}(N) = \frac{(2n+1)j+k+2n}{N} \geq \frac{N}{N} = 1$$

□

A.2. Visualización del Lema 1.16

Ejemplo A.4 (Ejemplo de aplicación del Lema 1.16)**.** Aplicamos el Lema con $\psi(x) = x^2$ y consecuentemente $1 \leq k \leq 5$. Escogemos $\varepsilon = 0{,}5$ y miramos cómo queda en la primera capa.

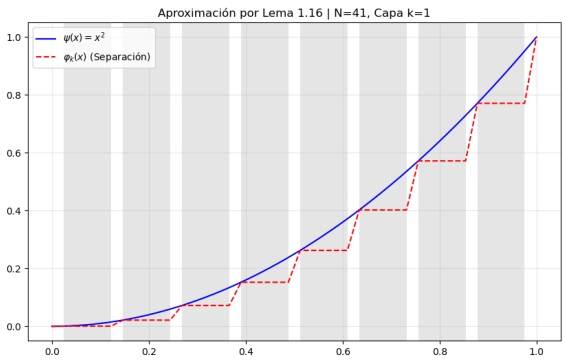

Figura A.1: Aplicación del Lema 1.16 para $\psi(x) = x^2$ y $\varepsilon = 0{,}5$.

A.2.1. Resultados y definiciones para la demostración del Teorema 1.12

Definición A.5 (Números linealmente independientes)**.** Sean $\kappa_1, \dots, \kappa_p$ números en $\mathbb{R}$. Decimos que son **linealmente independientes** sobre $\mathbb{Q}$ si y solo si se cumple que si $\lambda_i \in \mathbb{Q}$ y

$$\lambda_1\kappa_1 + \cdots + \lambda_p\kappa_p = 0,$$

entonces $\lambda_1 = \lambda_2 = \cdots = \lambda_p = 0$.

Proposición A.6. *Sean $p_1, p_2, \dots, p_n$ primos distintos. Definimos $\kappa_i = \sqrt{p_i}$. Entonces los números $\kappa_1, \dots, \kappa_n$ son linealmente independientes sobre $\mathbb{Q}$.*

La demostración de la Proposición A.6 puede encontrarse en el artículo Besicovitch [1940].

Teorema A.7 (Teorema de Baire)**.** *Sean E un espacio métrico completo y $(U_i)_{i\in\mathbb{N}}$ una sucesión de abiertos densos en E. Entonces $\bigcap_{i\in\mathbb{N}} U_i$ es denso en E.*

Lema A.8. *Sean $n \in \mathbb{N}^*$ y $\mathcal{K}$ el conjunto de matrices de Kolmogorov de dimensiones $(2n+1)\times n$ con $(\kappa_i)_{i=1}^n$ el conjunto fijado en la Proposición A.6. Entonces el espacio métrico $(\mathcal{K}, \|\cdot\|_\infty)$ es completo.*

Demostración. Sea $\{\Theta_k(t)\}_{k=1}^\infty \subset \mathcal{K}$ una sucesión de Cauchy, con $\Theta_k(t) = (\kappa_j\theta_{i,k}(t))_{i,j}$. Empezamos viendo que las sucesiones $\{\theta_{i,k}(t)\}_{k=1}^\infty$ son de Cauchy. Sea $\varepsilon > 0$ y $\kappa_{\text{mín}} = \text{mín}_{1\le j\le n}|\kappa_j| > 0$. Sabemos que existe $k_0 \in \mathbb{N}$ tal que si $p, q \ge k_0$, entonces

$$\|\Theta_p(t) - \Theta_q(t)\|_\infty < \varepsilon \cdot \kappa_{\text{mín}}.$$

Por la extensión de $\|\cdot\|_\infty$ a matrices de funciones, esto es equivalente a que

$$\max_{i,j} \|\kappa_j\theta_{i,p} - \kappa_j\theta_{i,q}\|_\infty < \varepsilon \cdot \kappa_{\text{mín}}.$$

De modo que para todo $p, q \ge k_0$ y $1 \le i \le 2n+1$, se tiene que

$$|\kappa_j|\|\theta_{i,p} - \theta_{i,q}\|_\infty \le \max_{i,j} \|\kappa_j\theta_{i,p} - \kappa_j\theta_{i,q}\|_\infty < \varepsilon \cdot \kappa_{\text{mín}}.$$

Lo que implica que para cualquier $1 \le j \le n$

$$\|\theta_{i,p} - \theta_{i,q}\|_\infty < \varepsilon \cdot \frac{\kappa_{\text{mín}}}{|\kappa_j|} \le \varepsilon.$$

Por consiguiente, las sucesiones $\{\theta_{i,k}(t)\}_{k=1}^\infty$ son de Cauchy. Por ser el espacio métrico $E = (C(I^1, \mathbb{R}), \|\cdot\|_\infty)$ completo, cada sucesión $\{\theta_{i,k}(t)\}_{k=1}^\infty$ converge a una función $\theta_i(t) \in E$.

Veamos que la sucesión original converge a la matriz de funciones $\Theta(t) = (\kappa_j\theta_i(t))_{i,j}$, que claramente está en $\mathcal{K}$. Sea $\varepsilon > 0$ y $\kappa_{\text{máx}} = \text{máx}_{1\le j\le n}|\kappa_j|$. Como $\theta_{i,k}(t)$ converge a $\theta_i(t)$, existe $k_{1,i} \in \mathbb{N}^*$ tal que si $p \ge k_{1,i}$ entonces $\|\theta_{i,p} - \theta_i\|_\infty < \frac{\varepsilon}{\kappa_{\text{máx}}}$. Escogiendo $k_1 = \text{máx}_i\, k_{1,i}$, se cumple la anterior desigualdad para todo i y $p \ge k_1$. Con lo cual,

$$\|\Theta_p(t) - \Theta(t)\|_\infty = \max_{i,j} \|\kappa_j\theta_{i,p} - \kappa_j\theta_i\|_\infty = \max_{i,j} |\kappa_j|\|\theta_{i,p} - \theta_i\|_\infty < \kappa_{\text{máx}} \frac{\varepsilon}{\kappa_{\text{máx}}} = \varepsilon.$$

Así que $\Theta_k(t)$ converge a $\Theta(t)$, lo que demuestra que $\mathcal{K}$ es completo. □

Teorema A.9 (Teorema de Aproximación de Weierstrass)**.** *Sea $f : I^n \to \mathbb{R}$ una función continua. Entonces, existe una sucesión de polinomios $(p_n(\mathbf{x}))_{n\in\mathbb{N}}$ con coeficientes en $\mathbb{Q}$ que converge uniformemente a la función $f(\mathbf{x})$.*

Proposición A.10. *Sea $(\Gamma_i(t))_{i=0}^\infty$ una sucesión de matrices de dimensión $1\times m$ tal que $\sum_{i=0}^\infty \Gamma_i(t)$ converge uniformemente a una matriz $\Gamma(t)$. Sea $B(t)$ una matriz de dimensión $m\times n$ y $\mathbb{D} \subset \mathbb{R}^n$ un compacto. Entonces*

$$\sum_{i=0}^{\infty} \Gamma_i(t) \star B(t) \star (\cdot) = \Gamma(t) \star B(t) \star (\cdot)$$

y su convergencia es uniforme sobre $\mathbb{D}$.

Demostración. Definimos $S_p(t) = \sum_{i=0}^{p} \Gamma_i(t)$ para $p \in \mathbb{N}$. Por la Proposición A.1,

$$\sum_{i=0}^{p} \big(\Gamma_i(t) \star B(t) \star (\cdot)\big) = S_p(t) \star B(t) \star (\cdot).$$

Basta pues con demostrar que $S_p(t) \star B(t) \star (\cdot)$ converge uniformemente a $\Gamma(t) \star B(t) \star (\cdot)$ sobre $\mathbb{D}$.

Escribimos $S_p(t) = (s_{1,k}(t))_{1\le k\le m}$ y $\Gamma(t) = (\gamma_{1,k}(t))_{1\le k\le m}$. Sea $\mathbf{x} \in \mathbb{D}$ y $\mathbf{y} = B(t) \star \mathbf{x} = (y_1, \ldots, y_m)$. Por la definición del operador $\star$,

$$\big|S_p(t) \star \mathbf{y} - \Gamma(t) \star \mathbf{y}\big| = \left|\sum_{k=1}^{m} \Big(s_{1,k}(y_k) - \gamma_{1,k}(y_k)\Big)\right| \le \sum_{k=1}^{m} |s_{1,k}(y_k) - \gamma_{1,k}(y_k)| \le m \, \|S_p(t) - \Gamma(t)\|_\infty.$$

Dado que el término de la derecha no depende de $\mathbf{x}$, se tiene que

$$\sup_{\mathbf{x}\in\mathbb{D}} \big|S_p(t) \star B(t) \star \mathbf{x} - \Gamma(t) \star B(t) \star \mathbf{x}\big| \le m \, \|S_p(t) - \Gamma(t)\|_\infty.$$

Dado que $\sum_{i=0}^{\infty} \Gamma_i(t)$ converge uniformemente a $\Gamma(t)$, entonces,

$$\sup_{\mathbf{x}\in\mathbb{D}} \big|S_p(t) \star B(t) \star \mathbf{x} - \Gamma(t) \star B(t) \star \mathbf{x}\big| \xrightarrow[p\to\infty]{} 0,$$

de modo que $S_p(t) \star B(t) \star (\cdot)$ converge uniformemente a $\Gamma(t) \star B(t) \star (\cdot)$ sobre $\mathbb{D}$, como queríamos demostrar. □

Apéndice B

Limitaciones del Teorema

B.1. Corolario del Teorema de Sternfeld

Corolario B.1. *Sean $n \geq 2$ y $m < 2n+1$. Entonces no existe ningún subconjunto de $\mathbb{R}^m$ que sea homeomorfo a I^n y básico.*

Demostración. El caso $m = 2n$ ya lo cubre el Teorema de Sternfeld 2.5. Podemos asumir pues que $m < 2n$. Argumentaremos por contradicción. Supongamos que existe un subconjunto $\mathbb{A} \subset \mathbb{R}^m$ que es homeomorfo a I^n y básico. Consideremos la inclusión de $\mathbb{R}^m$ sobre $\mathbb{R}^{2n}$ definida de la siguiente forma:

$$p_{m,2n} : \mathbb{A} \longrightarrow \mathbb{R}^{2n}$$

$$\mathbf{x}_m = (x_1, \ldots, x_m) \longmapsto p_{m,2n}(\mathbf{x}_m) = (x_1, x_2, \ldots, x_m, 0, \ldots, 0) \in \mathbb{R}^{2n}.$$

Si logramos ver que $p_{m,2n}(\mathbb{A}) \subset \mathbb{R}^{2n}$ es homeomorfo a I^n y básico habremos obtenido una contradicción con el Teorema de Sternfeld 2.5 y por ende quedará demostrado el corolario.

- **Homeomorfo a I^n:** Dado que la aplicación $p_{m,2n}$ es una inmersión, es un homeomorfismo sobre su imagen. Con lo cual tenemos

  $$I^n \cong \mathbb{A} \cong p_{m,2n}(\mathbb{A}).$$

  y queda claro que $p_{m,2n}(\mathbb{A})$ es un conjunto homeomorfo a I^n.

- **Básico:** Basta con ver que toda función de $p_{m,2n}(\mathbb{A})$ es simplemente representable. Sea pues $f(\mathbf{x}) : p_{m,2n}(\mathbb{A}) \to \mathbb{R}$ una función continua. Utilizando un razonamiento análogo al del ejemplo 2.3 definimos

  $$g : \mathbb{A} \longrightarrow \mathbb{R}$$

  $$\mathbf{x}_m = (x_1, \ldots, x_m) \longmapsto g(\mathbf{x}_m) = f(p_{m,2n}(\mathbf{x}_m)) = f(x_1, x_2, \ldots, x_m, 0, \ldots, 0).$$

  Lógicamente g es una función continua definida sobre un conjunto básico $\mathbb{A} \subset \mathbb{R}^m$, de modo que es simplemente representable, es decir, existe una matriz de funciones $A_m(t)$ de dimensión $1 \times m$ tal que

  $$A_m(t) \star \mathbf{x}_m = g(\mathbf{x}_m).$$

  Consideremos pues la matriz de funciones $A_{2n}(t) = \begin{pmatrix} A_m(t) & 0 & \ldots & 0 \end{pmatrix}$ de dimensión $1 \times 2n$. Entonces si $\mathbf{x} \in p_{m,2n}(\mathbb{A})$ se cumple que:

  $$A_{2n}(t) \star \mathbf{x} = A_m(t) \star p_{m,2n}^{-1}(\mathbf{x}) = A_m(t) \star \mathbf{x}_m = g(\mathbf{x}_m) = f(\mathbf{x}).$$

  Por tanto $f(\mathbf{x})$ es simplemente representable, por lo tanto, $p_{m,2n}(\mathbb{A})$ es básico.

Finalmente, deducimos que $p_{m,2n}(\mathbb{A})$ es un subconjunto de $\mathbb{R}^{2n}$ homeomorfo a I^n y básico, lo que entra en contradicción con el Teorema de Sternfeld 2.5. □

Apéndice C

Kolmogorov-Arnold Networks

C.1. Sobre las aristas

Algoritmo C.1 (Algoritmo de De Boor)**.** *Sea $t \in [t_k, t_{k+1})$ el punto de evaluación. Llamamos al subconjunto de $p+1$ puntos de control asociados a las funciones de base con t en su soporte como $d_j^{[0]} = c_j$, para $j = k-p, \ldots, k$.*

Para cada paso $r = 1, 2, \ldots, p$, y para cada $j = k-p+r, \ldots, k$, calculamos recursivamente:

$$d_j^{[r]} = (1 - \alpha_j^{[r]}) d_{j-1}^{[r-1]} + \alpha_j^{[r]} d_j^{[r-1]} \tag{C.1}$$

donde los pesos $\alpha_j^{[r]} \in [0, 1]$ se definen como:

$$\alpha_j^{[r]} = \frac{t - t_j}{t_{j+p-r+1} - t_j} \tag{C.2}$$

Tras completar los p pasos de la recursión, el valor final del B-Spline evaluado en t es exactamente el vértice del triángulo computacional:

$$r(t) = d_k^{[p]} \tag{C.3}$$

C.2. Ejemplo de actualización de atributos

Ejemplo C.2. Supongamos que tenemos una arista sin función base, únicamente con la familia residual de Chebyshev activada hasta grado 2. El dominio original está definido por $a = 0, b = 3$. La función aprendida por la arista es, por ejemplo, $a(t) = 3T_0(\hat{t}_{old}) + 2T_2(\hat{t}_{old})$, siendo el reescalado afín al dominio $[-1, 1]$ $\hat{t}_{old} = \frac{2}{3}t - 1$. Es decir:

$$a(t) = 3 + 2T_2\left(\frac{2}{3}t - 1\right)$$

Durante el entrenamiento, decidimos aumentar la expresividad añadiendo un nuevo grado (hasta grado 3) y actualizar el dominio a los nuevos extremos empíricos observados $a_{new} = -1, b_{new} = 4$. Supongamos que el lote de datos actual que atraviesa la arista es $\mathbb{D} = \{-1, -\frac{1}{2}, \frac{3}{2}, 4\}$.

1. **Evaluación empírica (Red antigua)**: Evaluamos $a(t)$ en cada punto de $\mathbb{D}$. Recordando que la definición matemática es $T_2(x) = 2x^2 - 1$, obtenemos:
   - $t = -1 \implies \hat{t}_{old} = -\frac{5}{3} \implies a(-1) = 3 + 2\left(2\left(-\frac{5}{3}\right)^2 - 1\right) = \frac{109}{9}$
   - $t = -\frac{1}{2} \implies \hat{t}_{old} = -\frac{4}{3} \implies a\left(-\frac{1}{2}\right) = 3 + 2\left(2\left(-\frac{4}{3}\right)^2 - 1\right) = \frac{73}{9}$
   - $t = \frac{3}{2} \implies \hat{t}_{old} = 0 \implies a\left(\frac{3}{2}\right) = 3 + 2(0 - 1) = 1 = \frac{9}{9}$
   - $t = 4 \implies \hat{t}_{old} = \frac{5}{3} \implies a(4) = 3 + 2\left(2\left(\frac{5}{3}\right)^2 - 1\right) = \frac{109}{9}$

Obtenemos así el vector de valores objetivo $\mathbf{y} = \left[\frac{109}{9}, \frac{73}{9}, 1, \frac{109}{9}\right]^T$.

2. **Generación de la nueva arista**: La nueva transformación afín es $\hat{t}_{new} = 2\frac{t-(-1)}{4-(-1)} - 1 = \frac{2}{5}t - \frac{3}{5}$. La nueva arista se define hasta grado 3, con los parámetros sin ajustar:

$$b(t) = c_0 T_0(\hat{t}_{new}) + c_1 T_1(\hat{t}_{new}) + c_2 T_2(\hat{t}_{new}) + c_3 T_3(\hat{t}_{new})$$

Si evaluamos la transformación $\hat{t}_{new}$ para los puntos reales de $\mathbb{D}$, obtenemos sus respectivos puntos internos mapeados en el espacio de Chebyshev: $\{-1, -\frac{4}{5}, 0, 1\}$.

3. **Planteamiento del sistema (Matriz de Diseño)**: Para plantear la proyección, evaluamos los cuatro polinomios de Chebyshev base (T_0, T_1, T_2, T_3) en los nuevos puntos internos mapeados, construyendo la matriz A:

$$A = \begin{pmatrix} 1 & -1 & 1 & -1 \\ 1 & -4/5 & 7/25 & 44/125 \\ 1 & 0 & -1 & 0 \\ 1 & 1 & 1 & 1 \end{pmatrix}$$

4. **Proyección de Parámetros**: Resolvemos el problema de regresión $A\mathbf{c} = \mathbf{y}$ para encontrar los coeficientes óptimos $\mathbf{c} = [c_0, c_1, c_2, c_3]^T$.

$$\begin{pmatrix} 1 & -1 & 1 & -1 \\ 1 & -4/5 & 7/25 & 44/125 \\ 1 & 0 & -1 & 0 \\ 1 & 1 & 1 & 1 \end{pmatrix} \begin{pmatrix} c_0 \\ c_1 \\ c_2 \\ c_3 \end{pmatrix} = \begin{pmatrix} 109/9 \\ 73/9 \\ 9/9 \\ 109/9 \end{pmatrix}$$

Al resolver este sistema (que en este caso tiene solución exacta por contar con 4 puntos para 4 parámetros), obtenemos:

$$c_0 = \frac{59}{9}, \quad c_1 = 0, \quad c_2 = \frac{50}{9}, \quad c_3 = 0$$

Apéndice D

Resultados

D.1. Configuración del ajuste general de funciones

Mostramos las configuraciones utilizadas para los experimentos mostrados en el Capítulo 5.

Elemento	**Configuración**
Epochs	2000
Tasa de aprendizaje	0,01
Atributos iniciales	$g = 5$, $p = 7$, $n_{\text{freq}} = 3$
Actualizaciones de atributos	2
Reescalado	Afín (Chebyshev y Fourier)

Tabla D.1: Configuración general del ajuste de funciones y de la comparativa de reescalados sobre esas mismas funciones. Los atributos finales de cada función se indican en la tabla del experimento correspondiente.

Parámetro	**Chebyshev**	**Fourier**
Epochs	2000	2000
Tasa de aprendizaje	0,01	0,01
Actualizaciones de atributos	$[7, 9, 12]$	$[3, 4, 6]$
N.º de actualizaciones	2	2
Reescalado	Afín / Global	Afín / Global

Tabla D.2: Configuración de los dos experimentos adicionales de la comparativa de reescalados.

Parámetro / Elemento	**Nuestra implementación**	**pykan**
Familia	B-splines	B-splines
Epochs (pre-poda)	2000	$[150, 105, 114]$
Tasa de aprendizaje	0,01	0,01
Actualizaciones de g	$[5, 7, 10]$	$[5, 7, 10]$
Iteraciones de refinamiento	2000 *epochs*	$[150, 105, 114]$ iter.
λ	0,01	0,01
$[\mu_1, \mu_2]$	[1,0; 1,0]	[1,0; 1,0]
Umbral de poda	**0,3**	**0,01**

Tabla D.3: Configuración e hiperparámetros de los experimentos de evaluación de la poda. El umbral no es comparable entre ambas implementaciones por emplear criterios de poda distintos (relativo frente a absoluto).

Parámetro	**Configuración**
Epochs (selección)	2000
Epochs (reentreno)	2000
Tasa de aprendizaje	0,01
λ	0,01
$[\mu_1, \mu_2]$	[1,0; 1,0]

Tabla D.4: Configuración e hiperparámetros de la fase de selección de familias.