UNIVERSITAT DE BARCELONA

Facultat de Matemàtiques
i Informàtica

GRAU DE MATEMÀTIQUES

Treball final de grau

Kolmogorov-Arnold representation theorem and its applications to neural networks

Autor: Miguel Sánchez-Palomino

Directors:	**Dr. Carles Rovira Escofet**	**(UB)**
	Dr. Patrick Aloy Calaf	**(IRB)**
	Dr. Bryan Percy Saldivar	**(IRB)**
Realitzat a:	**Institut de Recerca Biomèdica**	**(IRB)**

Barcelona, January, 2026

Acknowledgements

I would like to thank my supervisors both at IRB Barcelona and at UB for their attent supervision and valuable feedback. I am also grateful to these institutions by providing me both the knowledge and the opportunities required for the completion of this work. Finally, I want to acknowledge my family and friends by providing support and making easier the confection of this project.

Contents

Resum **v**

Abstract **vii**

Introduction **1**

1 Splines theory **3**
1.1 Splines theory . . . 3
1.1.1 Splines . . . 3
1.1.2 B-splines . . . 11

2 Kolmogorov-Arnold representation theorem **19**
2.1 Kolmogorov-Arnold representation theorem . . . 19

3 Neural networks **33**
3.1 Neural networks . . . 33
3.1.1 Objective . . . 33
3.1.2 Building blocks . . . 33
3.1.3 Forward pass . . . 35
3.1.4 Evaluation and training . . . 36
3.1.5 KANs . . . 40

Conclusion **48**

A Real analysis theorems **51**
A.1 Heine-Cantor theorem . . . 51
A.2 Weierstrass approximation theorems . . . 52

Bibliography **58**

Resum

Profundament arrelat en la humanitat trobem el desig d'entendre i dominar el nostre entorn, així impulsant el desenvolupament tant d'eines intel·lectuals com d'enginyeria per facilitar-nos aquesta missió. Fins ara, aquesta necessitat s'ha abordat mitjançant el desenvolupament de nous resultats matemàtics i les seves aplicacions a la modelització del món. Un dels nostres reptes més grans ha estat la creació de màquines que puguin pensar. Teoritzades molt abans de tenir la tecnologia per desenvolupar-les, finalment hem arribat als primers "cervells artificials" que poden veure més enllà del que mai nosaltres hauríem pogut.
En aquest projecte volem centrar-nos en una arquitectura particular que hi ha darrere d'un d'aquests "cervells". Basant-se en gran mesura en el teorema de representació de Kolmogorov-Arnold, les KANs semblen poder millorar algunes peces fonamentals de la IA. A més, son capaces de capturar explícitament relacions matemàtiques en les dades observades, la qual cosa les fa més interpretables. A causa de la seva estreta relació amb les matemàtiques i la bellesa que hi ha al darrere, les hem triat com a objecte d'estudi en aquest treball.

Abstract

Deeply rooted in humankind is the desire to understand and rule our surroundings, driving the development of both intellectual and engineering tools to ease us this mission. So far, this urge has been tackled through the development of new mathematical results, and its applications to the modelling of the world. One of our biggest challenges has been the creation of machines that can think. Theorized long before we had the technology to develop them, we've finally got to the first "artificial brains" that can see further than we ever could.
In this project we want to focus on one particular architecture behind one of those "brains". Relying heavily on the Kolmogorov-Arnold representation theorem, KANs seem capable of improving some building blocks of AI. Also they appear to explicitly capture mathematical relations on the observed data, thus making them more interpretable. Due to their close relation to mathematics and the beauty behind those, we have taken them to be the object of study in this work.

Introduction

In the year 1900 David Hilbert proposed a highly acclaimed list of problems. It contained 23 distinct open questions of mathematics, believing that the resolution of those would shape the XX-th century of mathematics. Of these, nine problems have been already solved, nine more have solutions with partial acceptance, three remain unresolved, and the remaining two have been deemed to be too vague to ever be resolved. We want to focus on the unresolved 13th problem.

The problem questions whether a solution exists to any seventh degree polynomial using only algebraic functions of two variables. Even if this is the original wording established by Hilbert, he later inquired about the existence of the solution in the class of continuous functions. The variant of this problem was solved in 1956 by the nineteen year old Vladimir Arnold under the tutelage of Andrey Kolmogorov (who published the result in 1957). In fact the proven theorem was fairly more general than the variation of Hilbert's problem, leaving its proof as a corollary. This theorem is known as *Kolmogorov-Arnold representation theorem (KAT or KART)*, and its reach highly exceeds the solution of Hilbert's problem as it will later become explicit.

During the same period, many well-known mathematicians, computer and cognitive scientists were establishing the basis of a new branch of science named Artificial Intelligence (AI). This new science was not something new, as since its beginning, humans had been in pursuit of an enhancement of their capabilities. Prior to the development of AI, all tools were improvements of physical traits, such as the hands, the legs or the sight; it was easy to theorize whether or not a machine could be develop to improve other non-physical features, in particular, if we could ever develop a machine capable of thinking on its own, and if so, would it ever surpass the intelligence of mankind?

Initially there were two currents of AI. The first considered that such machines would consist on the manipulation of logical symbols and they would derive conclusions on how to proceed based on the result. The other current attempted to model a human brain by creating networks of neurons. By strengthening or weakening the connection between them, the system would be able to decide. At the

time, given the limited computing power available to scientists, the first current was more prominent, but there were some theoretical advancements on the latter. When computing power was much more affordable, the superiority of neural networks was evident, and eventually these became ubiquitous in a myriad of different situations.

Originally the simplest neural networks (fully-connected forward-feeding networks) backed on variants of the universal approximation theorem. But given the generality of the KAT, it was not long before questions started emerging about an architecture backed on this theorem. Many of the early attempts failed when generalizing the network to deeper cases. This architecture was finally developed, and thoroughly studied by Liu et al. who in a 2024 paper provided full detail of this new model. They baptized it as Kolmogorov-Arnold networks (KANs).

In this work we aim to provide a complete and self-consistent recapitulation of the mathematical results that are necessary to understand the background of KANs. The work is structured in three chapters, one dedicated to splines and its most common basis, b-splines. Those are important as they play a fundamental role in the interpolation steps of KANs. The second chapter is dedicated to the Kolmogorov-Arnold representation theorem, where it is both enunciated and proved. For the proof we have followed the Dzhenzher and Skopenkov constructive proof, published in 2021, but we've explicitly proved the n case. The final chapter is dedicated to cover the construction and KANs and its comparison to multilayer perceptrons (MLPs) a building block of machine learning and artificial intelligence.

As we are conscious that the readers of this work might not be familiar with the jargon and proceedings of AI, so the last chapter centers on the simplest of procedures and overlooks some commonly-used sophistications to the algorithms and procedures. At the end of this chapter there are two simple examples that explicit the use of KANs for science and highlight their value.

Besides all that is explained on top, one can find a short appendix containing proofs to two theorems required for the proof of the Kolmogorov-Arnold representation theorem. Those are separated from the main body of the work as we assume the reader to be well-versed on them since they are covered in the syllabus of a typical bachelor in mathematics.

Chapter 1

Splines theory

1.1 Splines theory

1.1.1 Splines

Definition 1.1.1. Let an interval $[a, b] \subset \mathbb{R}$, consider $k+1$ ordered points t_i such that $a = t_0 \leq t_1 \leq \cdots \leq t_{k-1} \leq t_k = b$. Each of those $k+1$ points is called a **knot**. The non-decreasing vector $\mathbf{t} = (t_0, \ldots, t_k)$ is known as the **knot vector**. If all knots are equidistantly distributed within the interval, the knot vector is called **uniform**, otherwise it is called **non-uniform**.

Observation. For every two consecutive knots t_i and t_{i+1}, we can consider the interval $[t_i, t_{i+1})$, thus providing a disjoint cover of the interval $[a, b]$. I.e. $[a, b] = [t_0, t_1) \cup [t_1, t_2) \cup \cdots \cup [t_{k-1}, t_k) \cup [t_k]$

Definition 1.1.2 (Spline)**.** A spline $S_{\mathbf{t}}$ (with knot vector $\mathbf{t}$) is a piecewise polynomial function $S_{\mathbf{t}} : [a, b] \to \mathbb{R}$, such that each piece corresponds to a polynomial $P_i : [t_i, t_{i+1}) \to \mathbb{R}$, $i \in [0..k-1]$

Definition 1.1.3. A multidimensional parametric curve

$$\begin{array}{rccl} S: & [a,b] & \to & \mathbb{R} \times \mathbb{R} \times \cdots \times \mathbb{R} \\ & t & \mapsto & (X_0(t), X_1(t), \ldots, X_m(t)) \end{array}$$

is a spline if $X_i : [a, b] \to \mathbb{R}$ is a spline $\forall i = 0, \ldots, m$

Definition 1.1.4. The **degree** (respectively **order**) of a spline corresponds to the highest degree (resp. order) among its defining polynomials, that is $\deg(S_{\mathbf{t}}) = \max_i \{\deg(P_i)\}$ (resp. $\operatorname{ord}(S_{\mathbf{t}}) = \max_i \{\operatorname{ord}(P_i)\}$).

Observation. Typically all considered polynomials within a spline are taken to be of the same degree (resp. order), thus having $\deg(S_{\mathbf{t}}) = \deg(P_i)$ (resp. $\operatorname{ord}(S_{\mathbf{t}}) = \operatorname{ord}(P_i)$) for any value of i.

Definition 1.1.5. If $S \in \mathcal{C}^{r_i}$ in a neighbourhood of t_i, then the spline is said to be of smoothness $\mathcal{C}^{r_i}$. This means that both P_i and P_{i+1} share common values of the derivatives up to order r_i at t_i. The **smoothness vector** of a given knot vector $\mathbf{t}$ can be defined as $\mathbf{r_t} = (r_1, \ldots, r_{k-1})$. To ease the notation we can write $\mathbf{r}$ instead of $\mathbf{r_t}$. Alternatively, we can define the **smoothness loss** j_i of a knot t_i as $j_i = n - r_i$ where n is the degree of the spline (and also its maximum smoothness)

Definition 1.1.6. We denote by $S_n^{\mathbf{r}}(\mathbf{t})$ the set of all splines having knot vector $\mathbf{t}$, smoothness vector $\mathbf{r}$ (defined on knots $\mathbf{t}$) and degree $\leq n$. This set is known as the **spline space**.

Proposition 1.1.1. *If one knot t_{i+1} is taken to approach another knot t_i so that they become coincident, then the smoothness of the spline at such point is $n - j_i - j_{i+1}$. I.e. in a neighbourhood of the resulting knot t_i^* we have that the spline is of class $\mathcal{C}^{n-j_i-j_{i+1}}$*

Proof. Let the knots t_i and t_{i+1} have smoothness losses j_i and j_{i+1} respectively. After $t_{i+1} \to t_i$, the resulting knot t_i^* has a reduction of smoothness of $j_i + j_{i+1}$, then leaving the spline to be of smoothness $n - j_i - j_{i+1}$ or equivalently being of class $\mathcal{C}^{n-j_i-j_{i+1}}$ □

Definition 1.1.7. According to the above proposition, the spline to be characterized only by a knot vector $\mathbf{t}$ with knot repetitions. Each knot repetition corresponds to a loss of smoothness of one, then there is an equivalence between smoothness vector $\mathbf{r} = (r_1, \ldots r_{k-1})$ and an extended knot vector $\mathbf{t}_{ext}$ by doing

$$\mathbf{t}_{ext} = (t_0, \overbrace{t_1, \ldots, t_1}^{n-r_1=j_1}, \overbrace{t_2, \ldots, t_2}^{n-r_2=j_2}, t_3, \ldots, \overbrace{t_{k-1}, \ldots, t_{k-1}}^{n-r_{k-1}=j_{k-1}}, t_k)$$

Observation. In order to clamp the spline to the extremal knots, we need to repeat knots t_0 and t_k $n+1$ times, thus when referring to the extended knot vector $\mathbf{t}_{ext}$ we are referring to the non-decreasing vector

$$\mathbf{t}_{ext} = (\overbrace{t_0, \ldots, t_0}^{n+1}, \overbrace{t_1, \ldots, t_1}^{n-r_1}, t_2, \ldots, \overbrace{t_{k-1}, \ldots, t_{k-1}}^{n-r_{k-1}}, \overbrace{t_k, \ldots, t_k}^{n+1}) \tag{1.1}$$

Definition 1.1.8. Denoting by $\mathcal{F}$ the space of functions from an interval $[a, b]$ to the real numbers, i.e. $\mathcal{F} = \{f : [a, b] \to \mathbb{R}\}$. We can define **pointwise addition** as a binary operation between two functions $f, g : [a, b] \to \mathbb{R}$ as

$$+ : \begin{array}{ccc} \mathcal{F} \times \mathcal{F} & \to & \mathcal{F} \\ (f, g) & \mapsto & f + g \end{array} \quad \text{where} \quad f + g : \begin{array}{ccc} [a, b] & \to & \mathbb{R} \\ x & \mapsto & (f + g)(x) = f(x) + g(x) \end{array}$$

Definition 1.1.9. Considering $\mathcal{F}$, we can define product by a real scalar value as

$$\cdot : \begin{array}{ccc} \mathbb{R} \times \mathcal{F} & \to & \mathcal{F} \\ (\lambda, f) & \mapsto & \lambda \cdot f \end{array} \quad \text{where} \quad \lambda \cdot f : \begin{array}{ccc} [a, b] & \to & \mathbb{R} \\ x & \mapsto & (\lambda \cdot f)(x) = \lambda \cdot f(x) \end{array}$$

Proposition 1.1.2. *$S_n^{\mathbf{r}}(\mathbf{t})$ with the operations above is a vector space over $\mathbb{R}$*

Proof. In order to prove the above proposition, we need to prove the eight axioms of a vector space. Mostly those are inherited by the real number field $\mathbb{R}$, but explicitly we can prove it by doing

1. $\forall f, g, h \in S_n^{\mathbf{r}}(\mathbf{t})$, $f + (g + h) = (f + g) + h$

   Two functions are the same if their images for all points is the same, hence taking an arbitrary point $x \in [a, b]$.

   $[f + (g + h)](x) = f(x) + (g + h)(x) = f(x) + g(x) + h(x) = (f + g)(x) + h(x) = [(f + g) + h](x)$
2. $\forall f, g \in S_n^{\mathbf{r}}(\mathbf{t})$, $f + g = g + f$

   $(f + g)(x) = f(x) + g(x) = g(x) + f(x) = (g + f)(x)$, by commutativity of $\mathbb{R}$
3. $\exists 0 \in S_n^{\mathbf{r}}(\mathbf{t})$ such that $\forall f \in S_n^{\mathbf{r}}(\mathbf{t})$, $f + 0 = f$

   We can consider the function $0 : [a, b] \to \mathbb{R}$ such that $0(x) = 0 \forall x$. It is a spline of knot vector $\mathbf{t}$ (taking all defining polynomials constant to 0)

   Now, if $f \in S_n^{\mathbf{r}}(\mathbf{t})$, then $(f + 0)(x) = f(x) + 0(x) = f(x)$
4. $\forall f \in S_n^{\mathbf{r}}(\mathbf{t})$, $\exists -f \in S_n^{\mathbf{r}}(\mathbf{t})$ such that $f + (-f) = 0$

   Given a spline $f \in S_n^{\mathbf{r}}(\mathbf{t})$, consider $-f = (-1)\cdot$, this function is clearly a spline as neither the smoothness, the range nor the degree of the defining polynomials are changed.

   Now, $(f + (-f))(x) = f(x) + (-f)(x) = f(x) - f(x) = 0$, $\forall x \in [a, b]$
5. $\lambda(\mu f) = (\lambda\mu)f$, $\forall \lambda\mu \in \mathbb{R}$ and $\forall f \in S_n^{\mathbf{r}}(\mathbf{t})$

   $[\lambda(\mu f)](x) = \lambda[(\mu f)(x)] = \lambda(\mu f(x)) = \lambda\mu f(x) = (\lambda\mu)f(x)$
6. $1 \cdot f = f$

   $(1 \cdot f)(x) = 1 \cdot f(x) = f(x)$, $\forall x \in [a, b]$
7. $\lambda(f + g) = \lambda f + \lambda g$

   $[\lambda(f + g)](x) = \lambda(f + g)(x) = \lambda(f(x) + g(x)) = \lambda f(x) + \lambda g(x)$, $\forall x \in [a, b]$
8. $(\lambda + \mu)f = \lambda f + \mu f$

   $[(\lambda + \mu)f](x) = (\lambda + \mu)f(x) = \lambda f(x) + \mu f(x)$, $\forall x \in [a, b]$

□

Having now proved that the spline space $S_n^{\mathbf{r}}(\mathbf{t})$ is a vector space, the next natural step is finding the dimension of this space as well as a basis.
In order to determine a basis for the spline space $S_n^{\mathbf{r}}(\mathbf{t})$, we can restrict the basis of a larger space $S_n(\mathbf{t})$. This is the space of all polynomial piecewise functions of order $\leq n$ over the knots $\mathbf{t}$.

Observation. We can see that $S_n^{\mathbf{r}}(\mathbf{t}) \subset S_n(\mathbf{t})$ since any spline that is defined over knots $\mathbf{t}$ and has an explicitly defined smoothness vector $\mathbf{r}$ is composed of polynomial pieces (up to order n) on each of the intervals given by $\mathbf{t}$, but any function composed of polynomial pieces (up to order n) on those intervals belongs to $S_n(\mathbf{t})$

Proposition 1.1.3. *The dimension of $S_n(\mathbf{t})$ is $k(n+1)$*

Proof. Given a knot vector $\mathbf{t} = (t_0, \ldots, t_k)$, we have k intervals on where to define a polynomial of degree up to n. Restricting to each of those intervals, we can now consider the span of all polynomials of degree up to n, as being $a_0 + a_1 x + a_2 x^2 + \cdots + a_n x^n$, which allows us to consider the basis $1, x, x^2, \ldots, x^n$, hence we have $n+1$ independent vectors for the basis on k independent intervals, which yields a dimension of the vector space of $k(n+1)$. □

Proposition 1.1.4. *Given a smoothness vector $\mathbf{r} = (r_1, \ldots, r_{k-1})$, the dimension of $S_n^{\mathbf{r}}(\mathbf{t})$ is*

$$k(n+1) - \sum_{i=1}^{k-1} (r_i + 1)$$

Proof. We know from the previous observation and proposition that $S_n^{\mathbf{r}}(\mathbf{t}) \subset S_n(\mathbf{t})$ and also that $\dim(S_n(\mathbf{t})) = k(n+1)$.
The smoothness vector $\mathbf{r} = (r_1, \ldots, r_{k-1})$ imposes that a given spline $s \in S_n^{\mathbf{r}}(\mathbf{t})$ satisfies $s \in \mathcal{C}^{r_i}$ in a neighbourhood of t_i. If we denote s_i the polynomial piece that defines s in the interval $[t_i, t_i + 1)$ Alternatively, we can express this as the set of equations

$$\left\{ s_i^{(j)}(t_i) - s_{i-1}^{(j)}(t_i) = 0 \right\}, \ \forall j \in [0..r_i]$$

So for each knot, we have a set of $r_i + 1$ conditions (each corresponding to the loss of one degree of freedom). By subtracting those to the total number of elements in the more general basis we get the dimension of the space. That is

$$\dim(S_n^{\mathbf{r}}(\mathbf{t})) = k(n+1) - \sum_{i=1}^{k-1} (r_i + 1)$$

□

Notation. We denote by $disc_f(t_0)$ the value of the discontinuity that function f presents at point t_0, that is

$$disc_f(t_0) = \lim_{t \to t_0^+} f(t) - \lim_{t \to t_0^-} f(t)$$

Definition 1.1.10. Given a real number t and a non-negative integer p we can define the **truncated power function** as

$$(\cdot - t)_+^p : \begin{array}{ccc} \mathbb{R} & \to & \mathbb{R}_{\geq 0} \\ x & \mapsto & (x-t)_+^p = \begin{cases} (x-t)^p & \text{if} \quad x > t \\ 0 & \text{if} \quad x \leq t \end{cases} \end{array}$$

If $p = 0$, and $x \leq t$, the function is valued at 0^0, which we take set to 0. In other words, if $p = 0$, we recover a unit-high step function with discontinuity at t

Notation. When p is not explicitly written in the function, it is taken as $p = 1$

Observation. The truncated power function is continuous if $p > 0$ and its derivative is $D(\cdot - t)_+^p = \frac{d(\cdot - t)_+^p}{d\cdot} = p(\cdot - t)_+^{p-1}$, therefore we can see that the truncated power function is of class $\mathcal{C}^{p-1}$, with its p-th derivative having a finite discontinuity of size $p!$, that is $disc_{D^p(\cdot - t)_+^p}(t) = p!$.

Notation. In here we have denoted the p-th derivative of a function f as $D^p f$ to avoid too many superindices. This notation is completely equivalent to $f^{(p)}$, which will also be present on further sections of this work.

Definition 1.1.11. Given a set of points $\{t_0, \ldots, t_k\}$ and a natural number n, we can define the linear functionals λ_{ij} as follows

$$\lambda_{ij} f = \begin{cases} D^j f(t_0) & \text{if} \quad i = 0 \\ disc_{D^j f}(t_i) & \text{if} \quad i = 1, \ldots, k-1 \end{cases} \qquad \text{for } j = 0, \ldots, n$$

Definition 1.1.12. Given a set of points $\{t_0, \ldots, t_k\}$ and a natural number n, we can define the functions φ_{ij} as follows

$$\varphi_{ij} : \begin{array}{ccc} \mathbb{R} & \to & \mathbb{R} \\ x & \mapsto & \varphi_{ij}(x) = \begin{cases} \frac{(x-t_0)^j}{j!} & \text{if} \quad i = 0 \\ \frac{(x-t_i)_+^j}{j!} & \text{if} \quad i = 1, \ldots, k-1 \end{cases} \end{array} \qquad \text{for } j = 0, \ldots, n$$

Observation. If we consider the knot vector $\mathbf{t}$ to play the role of the set of points and the maximum degree of the polynomial pieces of the spline the one of n, we can observe that all functions φ_{ij} belong to the space $S_n(\mathbf{t})$. I.e. $\varphi_{ij} \in S_n(\mathbf{t})$.

Proposition 1.1.5. $\lambda_{ij}\varphi_{rs} = \delta_{ir}\delta_{js}$

Proof. By definition we have that the function $\varphi_{rs} \in \mathcal{C}^{s-1}$ with $disc_{D^s\varphi_{rs}}(t_r) = 1$ (given by its normalization) and continuous elsewhere. Also for $r > 1$ the function is zero up to t_r. With this considerations, we can separate into four cases

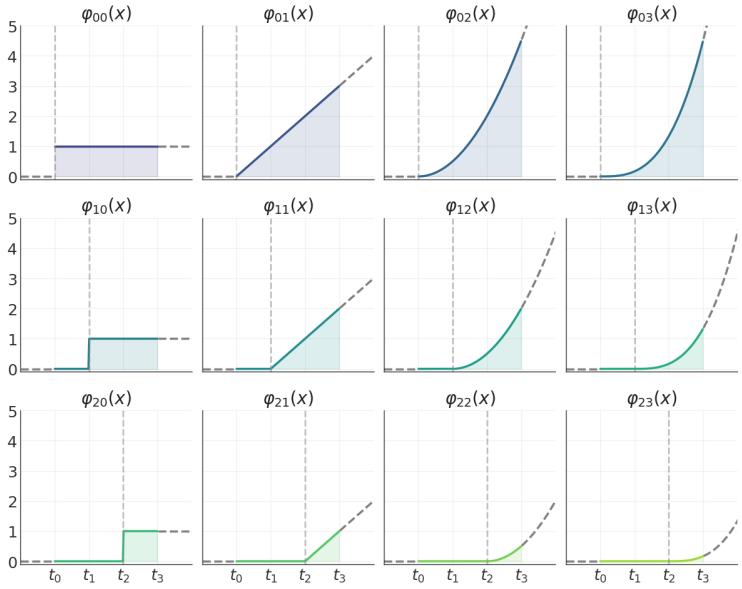


Figure 1.1: Truncated power functions φ_{ij} on a knot vector $\mathbf{t} = (t_0, t_1, t_2, t_3)$, up to degree $j = 3$. Dashed lines indicate the trajectory of the function outside the extremes of the knot vector

- $j > s$, we have $\lambda_{ij}\varphi_{rs} = 0$ as we are considering more derivatives than degree has the polynomial
- $j < s$, we have $\lambda_{ij}\varphi_{rs} = 0$ as the derivative we are considering is continuous. In case $i = 0$, the evaluation of the derivative at t_0 is either 0 (if $r > 0$) or th e expression evaluates to 0 (if $r = i = 0$)
- $i > r$, we have $\lambda_{ij}\varphi_{rs} = 0$ as the derivative we are considering can present at most one discontinuity at point t_r and we are considering points t_i where the function is continuous
- $i < r$, we have $\lambda_{ij}\varphi_{rs} = 0$ as all points to the left of t_r are 0 (by definition)

In the above cases we have proved the situations for $j > s$ and $j < s$ ($\forall i$ including $i = r$), and also the cases $i > r$ and $i < r$ ($\forall j$ including $j = s$). The remaining cases are only the ones where $j = s$ and $i = r$ simultaneously, hence we now study $\lambda_{ij}\varphi_{ij}$

$$\lambda_{ij}\varphi_{ij} = \begin{cases} \varphi_{ij}^{(j)}(t_0) & \text{if} \quad i = 0 \\ disc_{\varphi_{ij}^{(j)}}(t_i) & \text{if} \quad i > 0 \end{cases} = \begin{cases} \frac{d^j}{dt^j}\frac{(t-t_0)^j}{j!} & \text{if} \quad i = 0 \\ \lim\limits_{t \to t_i^+} \frac{d^j}{dt^j}\frac{(t-t_i)_+^j}{j!} - \lim\limits_{t \to t_i^-} \frac{d^j}{dt^j}\frac{(t-t_i)_+^j}{j!} & \text{if} \quad i > 0 \end{cases}$$

$$= \begin{cases} 1 & \text{if} \quad i = 0 \\ \lim\limits_{t \to t_i^+} 1 - \lim\limits_{t \to t_i^-} 0 & \text{if} \quad i > 0 \end{cases} = 1$$

Thus we get the sought result $\lambda_{ij}\varphi_{rs} = \delta_{ir}\delta_{js}$ □

Proposition 1.1.6. *The set* $\{\varphi_{ij}\}$ *is a basis of* $S_n(\mathbf{t})$

Proof. On proposition 1.1.3, we have seen that the dimension of $S_n(\mathbf{t})$ is $k(n+1)$, and we can see that there are also $k(n+1)$ functions defined in 1.1.12, and by proposition 1.1.5, those are independent, hence forming a basis of the space.
To see the independence of the functions we can prove it by assuming a linear combination

$$\sum_{i=0}^{k-1}\sum_{j=0}^{n} c_{ij}\varphi_{ij} = 0$$

Applying the functional λ_{rs} to both sides we have

$$\lambda_{rs}\left(\sum_{i=0}^{k-1}\sum_{j=0}^{n} c_{ij}\varphi_{ij}\right) = \sum_{i=0}^{k-1}\sum_{j=0}^{n} c_{ij}\lambda_{rs}\left(\varphi_{ij}\right) = \sum_{i=0}^{k-1}\sum_{j=0}^{n} c_{ij}\delta_{ir}\delta_{js} = c_{rs} = 0$$

As the choice of indexes r and s is arbitrary we can conclude that all coefficients c_{ij} need to be zero, hence proving the independence of $\{\varphi_{ij}\}$ □

Proposition 1.1.7. *Any spline* $s \in S_n(\mathbf{t})$ *has a unique representation*

$$s = \sum_{i=0}^{k-1}\sum_{j=0}^{n} \left(\lambda_{ij}s\right)\varphi_{ij} \tag{1.2}$$

Proof. Let $s \in S_n(\mathbf{t})$, we denote by s_i the polynomial piece that defines s compressed within the interval $[t_i, t_{i+1})$. We can expand each s_i by means of its Taylor series around point $t = t_i$, yielding

$$s_i = \sum_{j=0}^{n} \frac{s^{(j)}}{j!}(t - t_i)^j$$

And more generally we can exactly (as the degree can not exceed n) expand the whole function s as

$$s = \sum_{i=0}^{k-1} \chi_{[t_i,t_i+1)}(t) \sum_{j=0}^{n} \frac{s^{(j)}}{j!}(t - t_i)^j$$

Where $\chi_{[t_i,t_{i+1})}$ denotes the characteristic function of a set. I.e.

$$\begin{array}{rccl} \chi_{[t_i,t_{i+1}]}: & \mathbb{R} & \to & \{0,1\} \\ & t & \mapsto & \chi_{[t_i,t_{i+1})}(t) = \begin{cases} 1 & \text{if } t \in [t_i, t_{i+1}) \\ 0 & \text{elsewhere} \end{cases} \end{array}$$

We can now manipulate the expression of 1.2 as follows

$$s = \sum_{i=0}^{k-1}\sum_{j=0}^{n}\left(\lambda_{ij}s\right)\varphi_{ij} = \sum_{j=0}^{n} s^{(j)}(t_0)\varphi_{0j} + \sum_{i=1}^{k-1}\sum_{j=0}^{n} disc_{s^{(j)}}(t_i)\varphi_{ij}$$

$$= \sum_{j=0}^{n} s^{(j)}(t_0)\frac{(t-t_0)^j}{j!} + \sum_{i=1}^{k-1}\sum_{j=0}^{n} disc_{s^{(j)}}(t_i)\frac{(t-t_i)_+^j}{j!}$$

Now we can see that this expression is close to the desired one, but for a scaling factor $disc_{s^{(j)}}(t_i)$ instead of $s^{(j)}(t_i)$ in the second term. We can understand this by recalling that the basis functions are not restricted to the interval $[t_i, t_{i+1})$, but instead are defined for all $t \geq t_i$, thus when performing the expansion, we need to account for the previous expansion, as it is overlapping into the considered interval. In other words, we have that the Taylor expansion of s_i overlaps the interval of definition of s_{i+1}, and therefore the expansion for the latter should "remove" the contribution of the former.
Mathematically, by substituting the definition of $disc_f(t)$ onto **??** and observing that on the first term it is equivalent either having s_0 or s (since we are only considering the derivative at $t = t_0$, where s is defined by s_0), we get

$$s = \sum_{j=0}^{n} s_0^{(j)}(t_0)\frac{(t-t_0)^j}{j!} + \sum_{i=1}^{k-1}\sum_{j=0}^{n}\left(\left(\lim_{t\to t_i+} s^{(j)}(t) - \lim_{t\to t_i-} s^{(j)}(t)\right)\frac{(t-t_i)_+^j}{j!}\right)$$

$$= \sum_{j=0}^{n} s_0^{(j)}(t_0)\frac{(t-t_0)^j}{j!} + \sum_{i=1}^{k-1}\sum_{j=0}^{n}\left(\left(s_i^{(j)}(t_i) - s_{i-1}^{(j)}(t_i)\right)\frac{(t-t_i)_+^j}{j!}\right)$$

$$= \sum_{j=0}^{n} s_0^{(j)}(t_0)\frac{(t-t_0)^j}{j!} + \sum_{i=1}^{k-1}\sum_{j=0}^{n} s_i^{(j)}(t_i)\frac{(t-t_i)_+^j}{j!} - \sum_{i=1}^{k-1}\sum_{j=0}^{n} s_{i-1}^{(j)}(t_i)\frac{(t-t_i)_+^j}{j!}$$

$$= \sum_{i=0}^{k-1}\sum_{j=0}^{n} s_i^{(j)}(t_i)\frac{(t-t_i)_+^j}{j!} - \sum_{i=1}^{k-1}\sum_{j=0}^{n} s_{i-1}^{(j)}(t_i)\frac{(t-t_i)_+^j}{j!}$$

Where we can see that the first term for each i we do the Taylor expansion of s_i, while the second term removes the contribution that the Taylor expansion of s_{i-1} has on the points $t \geq t_i$.
By the unicity of the Taylor expansion, we have that this expansion of function s is also unique. □

Proposition 1.1.8. *$\{\varphi_{ij_i}\}$ with $i = 0, \ldots, k-1$ and $j_i = r_i, \ldots, n$, where r_i is the i-th component of the smoothness vector $\mathbf{r}$, and $r_0 = 0$, is a basis of the space $S_n^{\mathbf{r}}(\mathbf{t})$*

Proof. Let $s \in S_n^{\mathbf{r}}(\mathbf{t})$, the presence of a smoothness vector $\mathbf{r} = (r_1, \ldots, r_{k-1})$ imposes that $s \in \mathcal{C}^{r_i}$ at a neighbourhood of t_i. Alternatively, we can rewrite this condition using the definition of disc as

$$\lim_{t \to t_i^+} s^{(j)}(t) - \lim_{t \to t_i^-} s^{(j)}(t) = disc_{s^{(j)}}(t_i) = 0,\ i = 1, \ldots, k-1,\ j = 0, \ldots, r_i$$

Now we can see that using proposition 1.1.7, we can express any function $\tilde{s} \in S_n(\mathbf{t})$ as

$$\tilde{s} = \sum_{j=0}^{n} \tilde{s}^{(j)}(t_0)\varphi_{0j} + \sum_{i=1}^{k-1} \sum_{j=0}^{n} disc_{\tilde{s}^{(j)}}(t_i)\varphi_{ij}$$

And since $S_n^{\mathbf{r}}(\mathbf{t}) \subset S_n(\mathbf{t})$ we can express the function s in the above way. In this case, by equation 1.1.1 we can observe that for a given i the functions φ_{ij} with $j = 0, \ldots, r_i$ do not play any role in the expansion, hence we have enough considering the remaining

$$k(n+1) - \sum_{i=1}^{k-1}(r_i + 1)$$

independent functions. As the number of independent functions that we have considered is equal to the dimension of $S_n^{\mathbf{r}}(\mathbf{t})$ (proposition 1.1.4), we can state that $\{\varphi_{ij}\}$ with $i = 0, \ldots, k-1$, $j = r_i + 1, \ldots n$ is a basis of $S_n^{\mathbf{r}}(\mathbf{t})$. □

1.1.2 B-splines

There are a few reasons why the choice of basis defined above is not the optimal one. One of the limitations of this basis comes due to the non-locality of its functions, that means that a slight change on a coefficient can have a large impact throughout the whole representation to the right.
One solution to this issue is the adoption of B-splines functions (short for Basis splines functions) as a basis of $S_n^{\mathbf{r}}(\mathbf{t})$. This functions are defined so that they influence their neighbourhood as little as possible. Also those are much more convenient when approximating functions through splines. But have the downside of not passing through the points when doing an interpolation.

Definition 1.1.13. Given a maximum degree n, and an extended knot vector

$$\mathbf{t}_{ext} = (\overbrace{t_0, \ldots, t_0}^{n+1}, \overbrace{t_1, \ldots, t_1}^{n+1-r_1}, t_2, \ldots, \overbrace{t_{k-1}, \ldots, t_{k-1}}^{n+1-r_{k-1}}, \overbrace{t_k, \ldots, t_k}^{n+1}) = (t_0, \ldots, t_N)$$

we define the $i-th$ B-spline of degree p using the Cox-de Boor recursion formula

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

where $B_{i,0}(t)$ are defined as the characteristic function of the interval $[t_i, t_{i+1})$

$$B_{i,0}(t) = \chi_{[t_i,t_{i+1})}(t) = \begin{cases} 1 & \text{if } t \in [t_i, t_{i+1}) \\ 0 & \text{elsewhere} \end{cases}$$

Notation. In here we have made the abuse of language to call both the components of the "original" knot vector and the extended knot vector by t_i. Since on this section we will work only with the extended knot vector, we understand t_i to refer to the components of $\mathbf{t}_{ext}$. Being N the length of this one.

Observation. One can find a representation of a family of b-splines in figure 1.2

Proposition 1.1.9. *Given $\mathbf{t}_{ext} = (t_0, \ldots, t_N)$, there are $N - p$ splines of degree p*

Proof. We can prove this statement by observing the recursion formula.
By definition we have N splines of degree $p = 0$ $(B_{0,0}, \ldots, B_{N-1,0})$. The recursion formula for $B_{i,p}$ requires the definition of $B_{i+1,p-1}$.
Because of that, for $p = 0$, i takes the values $i \in [0..N-1]$ and for $p = 1$ we can only define up to $i = N - 2$. Continuing with this argument, we can observe that for degree $p = p$ we can only consider up to $i = N - (p + 1)$, hence leaving us with $N - p$ splines $(B_{0,p}, \ldots, B_{N-(p+1),p})$ □

Observation. We can infer from this proof that in order to construct a given b-spline we need to compute all the b-splines on top of it following a triangular scheme. This is illustrated in figure 1.2

Definition 1.1.14. The **support** of a function $f : X \to Y$ is the set $supp(f) = \{x \in X : f(x) \neq 0\}$

Proposition 1.1.10. $supp(B_{i,n}) \subseteq [t_i, t_{i+n+1})$

Proof. We can see by the recursion formula that a given b-spline $B_{i,n}$ is defined from $B_{i,n-1}$ and $B_{i+1,n-1}$, applying this process recursively to all splines allows us to state that the b-spline $B_{i,n}$ is generated from $B_{i,0}, B_{i+1,0}, \ldots, B_{i+n,0}$.
By definition the b-splines $B_{l,0}$ satisfy $supp\,(B_{l,0}) = [t_l, t_{l+1})$, then for our case we know that

$$\begin{aligned} supp(B_{i,n}) &\subseteq supp(B_{i,0}) \cup supp(B_{i+1,0}) \cup \cdots \cup supp(B_{i+n,0}) \\ &= [t_i, t_{i+1}) \cup [t_{i+1}, t_{i+2}) \cup \cdots \cup [t_{i+n}, t_{i+n+1}) = [t_i, t_{i+n+1}) \end{aligned}$$

□

Proposition 1.1.11. *Given a spline*

$$s(t) = \sum_{i=0}^{N-(n+1)} c_i B_{i,n}(t)$$

Where c_i are coefficients independent of t. For any point $x \in [t_0, t_N]$, the image $s(x)$ is determined only by $n+1$ b-splines

Proof. For any point $x \in [t_0, t_N]$, we can find an index $\mu \in [n..N-n]$ such that $x \in [t_\mu, t_{\mu+1})$. Also, by proposition 1.1.10, we know that $supp(B_{i,n}) \subseteq [t_i, t_{i+n+1})$, that can be understood as only b-splines that satisfy $[t_\mu, t_{\mu+1}) \subseteq supp(B_{i,n})$ are going to contribute to the image of $s(x)$ (as all other have support zero at the studied interval).
That condition translates to

$$[t_\mu, t_{\mu+1}) \subseteq [t_i, t_{i+n+1}]$$

or equivalently

$$\begin{cases} \mu \geq i \\ \mu + 1 \leq i + n + 1 \end{cases} \iff \begin{cases} i \leq \mu \\ i \geq \mu - n \end{cases}$$

Which are satisfied for values $i = \mu, \mu-1, \ldots, \mu-n$, so that the only b-splines that influence the image $s(t)$ are $B_{\mu,n}, B_{\mu-1,n}, \ldots, B_{\mu-n}$ (amounting to $n+1$ b-splines) □

Proposition 1.1.12. *Given a spline*

$$s(t) = \sum_{i=\mu-n}^{\mu} c_i B_{i,n}(t)$$

with known weights c_i, we can find the value of the spline evaluated at a point t as the value of c_μ^n defined by the recursion

$$c_i^d = \frac{x - t_i}{t_{i+n-d+1} - t_i} c_i^{d-1} + \frac{t_{i+n-d+1} - t}{t_{i+n-d+1} - t_i} c_{i-1}^{d-1}$$

where $c_i^0 = c_i$, $i \in [\mu - n..\mu]$

Proof. In order to prove this, we can rewrite the spline $s(t)$ using the Cox-de Boor recursion (equation 1.1.13) and rewrite the coefficients c_i as c_i^0, by doing this we have

$$s(t) = \sum_{i=\mu-n}^{\mu} c_i B_{i,n}(t) = \sum_{i=\mu-n}^{\mu} c_i^0 \left(\frac{t - t_i}{t_{i+n} - t} B_{i,n-1}(t) + \frac{t_{i+n+1} - t}{t_{i+n+1} - t_{i+1}} B_{i+1,n-1}(t) \right)$$

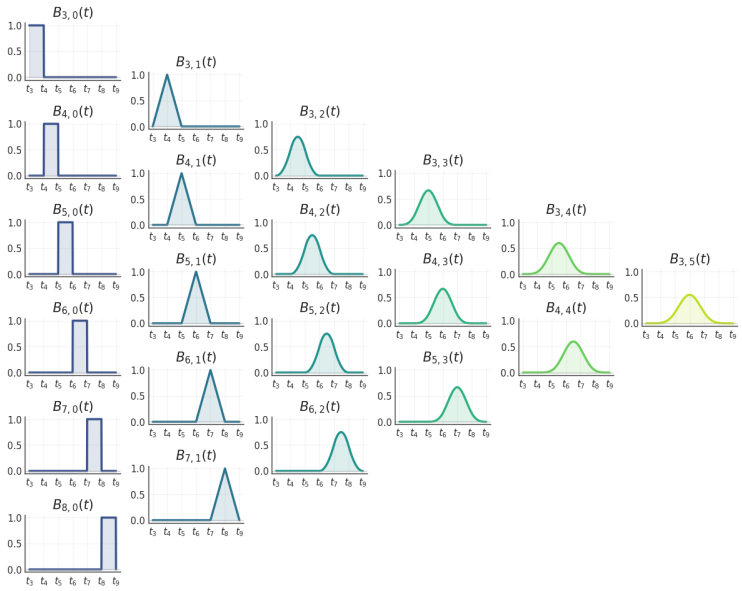


Figure 1.2: B-spline functions involved in the construction of $B_{3,5}(t)$ with an extended knot vector $\mathbf{t} = (t_0, t_0, t_0, t_0, t_1, t_2, t_3, t_4, t_5, t_6, t_6, t_6, t_6) = (t_0, \ldots, t_{12})$

$$= \left(c^0_{\mu-n} \frac{t - t_{\mu-n}}{t_\mu - t_{\mu-n}} B_{\mu-n,n-1}(t) + \frac{t_{\mu+1} - t}{t_{\mu+1} - t_{\mu-n+1}} B_{\mu-n+1,n-1}(t) \right) +$$

$$+ \left(c^0_{\mu-n+1} \frac{t - t_{\mu-n+1}}{t_{\mu+1} - t_{\mu-n+1}} B_{\mu-n+1,n-1}(t) + \frac{t_{\mu+2} - t}{t_{\mu+2} - t_{\mu-n+2}} B_{\mu-n+2,n-1}(t) \right) + \cdots +$$

$$+ \left(c^0_{\mu} \frac{t - t_{\mu}}{t_{\mu+n} - t_{\mu}} B_{\mu,n-1}(t) + \frac{t_{\mu+n+1} - t}{t_{\mu+n+1} - t_{\mu+1}} B_{\mu+1,n-1}(t) \right)$$

$$= c^0_{\mu-n} \frac{t - t_{\mu-n}}{t_\mu - t_{\mu-n}} B_{\mu-n,n-1}(x) + \left[\sum_{i=\mu-d+1}^{\mu} \left(\frac{t_{i+n} - t}{t_{+n} - t_i} c^0_{i-1} + \frac{t - t_i}{t_{i+n} - t_i} c^0_i \right) B_{i,n-1}(t) \right]$$

$$+ c^0_{\mu} \frac{t_{\mu+d+1} - t}{t_{\mu+d+1} - t_{\mu+1}} B_{\mu+1,n-1}(t)$$

And we have that both the first and last terms of this expression are zero by proposition 1.1.10. Hence we have

$$s(t) = \sum_{i=\mu-d+1}^{\mu} \left(\frac{t_{i+n} - t}{t_{+n} - t_i} c^0_{i-1} + \frac{t - t_i}{t_{i+n} - t_i} c^0_i \right) B_{i,n-1}(t) = \sum_{i=\mu-d+1}^{\mu} c^1_i B_{i,n-1}(t)$$

Where we have used the recursion for c^d_i on the wording. This illustrates that the

process allows us to reduce the summation indices by one, hence after d iterations we get

$$s(t) = \sum_{i=\mu-n+d}^{\mu} c_i^d B_{i,n-d}(t)$$

And in particular for $d = n$ we get

$$s(x) = c_\mu^d = c_\mu^n$$

□

Theorem 1.1.1 (Marsden's identity)**.** *For any $t \in [t_0, t_N]$ and for any $x \in \mathbb{R}$*

$$(x-t)^n = \sum_{i=0}^{N-(n+1)} \Psi_{i,n}(x) B_{i,n}(t)$$

Where

$$\Psi_{i,n}(x) = (x-t_{i+1})(x-t_{i+2})\cdots(x-t_{i+n}) = \prod_{j=1}^{n}(x-t_{i+j})$$

Proof. Since we are in the hypothesis of proposition 1.1.11, a proof for the local form of the theorem

$$(x-t)^n = \sum_{i=\mu-n}^{\mu} \Psi_{i,n}(x) B_{i,n}(t)$$

would suffice.
We can make use of proposition 1.1.12, and initialize the c_i^0 coefficients as $c_i^0 = \Psi_{i,n}(x)$, $i \in [\mu - d..\mu]$ (this can be done since the coefficients are not dependent on t).
Now, we can consider the recursion on coefficients given by the proposition 1.1.12,

$$\begin{aligned}
c_i^1 &= \frac{t-t_i}{t_{i+n}-t_i} c_i^0 + \frac{t_{i+n}-t}{t_{i+n}-t_i} c_{i-1}^0 = \frac{t-t_i}{t_{i+n}-t_i}\Psi_{i,n}(x) + \frac{t_{i+n}-t}{t_{i+n}-t_i}\Psi_{i-1,n}(x) \\
&= \frac{t-t_i}{t_{i+n}-t_i}\prod_{j=1}^{n}(x-t_{i+j}) + \frac{t_{i+n}-t}{t_{i+n}-t_i}\prod_{j=1}^{n}(x-t_{i-1+j}) \\
&= \left(\frac{t-t_i}{t_{i+n}-t_i}(x-t_{i+n}) + \frac{t_{i+n}-t}{t_{i+n}-t_i}(x-t_i)\right)\prod_{j=1}^{n-1}(x-t_{i+j}) \\
&= \left(\frac{t-t_i}{t_{i+n}-t_i}(x-t_{i+n}) + \frac{t_{i+n}-t}{t_{i+n}-t_i}(x-t_i)\right)\Psi_{i,n-1}(x)
\end{aligned}$$

And we can now modify the expression between parenthesis as

$$\frac{t-t_i}{t_{i+n}-t_i}(x-t_{i+n})+\frac{t_{i+n}-t}{t_{i+n}-t_i}(x-t_i)=\frac{-t_ix-tt_{i+n}+t_{i+n}x+tt_i}{t_{i+n}-t_i}=x-t$$

So we can express it as

$$c_i^1=(x-t)\Psi_{i,n-1}(x)$$

Repeating the reasoning we can see that

$$c_i^d=(x-t)^d\Psi_{i,n-d}(x)\qquad\text{and eventually}\qquad c_i^n=(x-t)^n\Psi_{i,0}(x)=(x-t)^n$$

And using proposition 1.1.12 once more we have what we wanted to prove as $c_i^n=s(x)=(x-t)^n$ □

Proposition 1.1.13. *All splines of degree n ($B_{0,n}, B_{1,n}, \ldots, B_{N-(n+1),n}$) are linearly independent in the interval $[t_0,t_N]=[t_n,t_{N-n}]$*

Proof. We can prove that the above b-splines are independent showing that if

$$\sum_{i=0}^{N-(n+1)}c_iB_{i,n}(t)=0$$

for any value $t\in[t_0,t_N]$, then $c_i=0$ for all values of $i\in[0..N-(n+1)]$
Once again, we can study the simplified problem

$$\sum_{i=\mu-n}^{\mu}c_iB_{i,n}(t)=0 \tag{1.3}$$

with $t_\mu\le t\le t_{\mu+1}$, where we want to see that $c_{\mu-n}=\cdots=c_\mu=0$. Since there are only $n+1$ b-splines at play ($B_{\mu-n,n}, B_{\mu-n+1,n}, \ldots, B_\mu$), it is enough to express all monomials t^d with $0\le d\le n$ as a linear combination of b-splines. Differentiating $n-d$ times Marsden's identity (theorem 1.1.1) with respect to x, we get

$$\frac{d^{n-d}}{dx^{n-d}}\left[(x-t)^n=\sum_{i=\mu-n}^{\mu}\Psi_{i,n}(x)B_{i,n}(t)\right]=\left[\frac{n!}{d!}(x-t)^d=\sum_{i=\mu-n}^{\mu}\frac{d^{n-d}}{dx^{n-d}}\Psi_{i,n}(x)B_{i,n}(t)\right]$$

By setting $x=0$, Marsden's identity becomes

$$\frac{n!}{d!}(-t)^d=\sum_{i=\mu-n}^{\mu}\left(\frac{d^{n-d}}{dx^{n-d}}\Psi_{i,n}(x)\bigg|_{x=0}\right)B_{i,n}(t)$$

Solving for t^d, we arrive to

$$t^d=\frac{d!}{n!}(-1)^d\sum_{i=\mu-n}^{\mu}\left(\frac{d^{n-d}}{dx^{n-d}}\Psi_{i,n}(x)\bigg|_{x=0}\right)B_{i,n}(t)$$

and comparing with the expression 1.3, we get the expression for the coefficients

$$c_i = \frac{d!}{n!}(-1)^d \left.\frac{d^{n-d}}{dx^{n-d}}\Psi_{i,n}(x)\right|_{x=0}$$

This way we have proved that all b-splines $B_{\mu-n,n}, \ldots, B_{\mu,n}$ are independent in the interval $[t_\mu, t_\mu + 1)$ thus imposing that $c_{\mu-n} = \cdots = c_\mu = 0$, repeating this arguments for all values of μ we have that all b-splines are independent. That follows from the fact that a different value of μ' yields b-splines that are either equal or with different support to the ones in μ. For the latter case, that means that necessarily those are independent from the ones in μ. □

Proposition 1.1.14. *B-splines* $\{B_{0,n}, \ldots B_{N-(n+1),n}\}$ *are an alternative basis to* $S_n^{\mathbf{r}}(\mathbf{t})$

Proof. Note that in here we denote by t_i^{ext} the knots from the extended knot vector, and by t_i the ones from the knot vector.
We have already seen that all the mentioned b-splines are independent (proposition 1.1.13), on the one hand we recall from proposition 1.1.4 that

$$dim(S_n^{\mathbf{r}}(\mathbf{t})) = k(n+1) - \sum_{i=1}^{k-1}(r_i+1) = k(n+1) - (k-1) - \sum_{i=1}^{k-1} r_i = kn + 1 - \sum_{i=1}^{k-1} r_i$$

where k is the number of knots of the knot vector, and n the degree of the spline. On the other hand we have $N - n$ independent b-splines, where N is the number of knots in the extended vector, which observing 1.1 we observe that

$$N = (n+1) + (n - r_1) + \cdots + (n - r_{k-1}) + (n+1) - 1 = n(k+1) + 1 - \sum_{i=1}^{k-1} r_i$$

and by subtracting n we get

$$N - n = n(k+1) + 1 - \sum_{i=1}^{k-1} r_i - n = nk + 1 - \sum_{i=1}^{k-1} r_i$$

Since we have the same number of functions as the dimension and those are independent and contained in the space, we have that the b-splines form a basis of the space. □

Observation. Any spline can be represented as

$$s(t) = \sum_{i=0}^{N-(n+1)} c_i B_{i,n}(t)$$

this result will prove particularly useful when trying to interpolate datapoints

Chapter 2

Kolmogorov-Arnold representation theorem

2.1 Kolmogorov-Arnold representation theorem

In 1957, on his paper *On the representation of continuous functions of several variables as superpositions of continuous functions of one variable and addition*[1], Andrey Kolmogorov proved that any continuous function $f : [0,1]^n \to \mathbb{R}$ could be represented using only addition and continuous functions of one variable. This work was constructed on top of simultaneous and previous statements by Vladimir Arnold, this result proved Hilbert's thirteenth problem as a corollary.
The original statement is as follows

Theorem 2.1.1 (Kolmogorov-Arnold representation)**.** *For any integer $n \geq 2$ and any continuous function $f(x_1, \ldots, x_n) : [0,1]^n \to \mathbb{R}$, exist continuous functions $\varphi_{pq} : [0,1] \to \mathbb{R}$ and $\Phi_q : \mathbb{R} \to \mathbb{R}$ such that we have the representation*

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

In order to prove this theorem we will prove a more general version, which can later be relaxed to obtain this result (following the proof of Dzhenzher and Skopenkov[7]). We assume the reader is familiar with some real analysis theorems such as Weierstrass approximation or Heine-Cantor, for detailed proofs of these, see appendix A. The theorem we intend to prove is the following

Theorem 2.1.2. *For any integer $n \geq 2$ and any continuous function $f(x_1, \ldots, x_n) : [0,1]^n \to \mathbb{R}$, exist real numbers $\alpha_1, \ldots, \alpha_n \in \mathbb{R}$, exist continuous functions $\varphi_1, \ldots, \varphi_{2n+1} :$*

$[0,1] \to [0,1]$ *and* $h : \mathbb{R} \to \mathbb{R}$ *such that we have the representation*

$$f(x_1, \ldots, x_n) = \sum_{k=1}^{2n+1} h\left(\sum_{i=1}^{n} \alpha_i \varphi_k(x_i)\right)$$

We can now prove that theorem 2.1.2 implies 2.1.1 as follows

Proposition 2.1.1. *Theorem 2.1.2 implies 2.1.1*

Proof. We can consider the functions Φ_q to be equal to h for all $q \in [1..2n+1]$, since both are continuous and have the same domain and codomain.
Also we can define the functions $\varphi_{q,p}$ as

$$\begin{array}{rccl} \varphi_{q,p}: & [0,1] & \to & [0,1] \\ & t & \mapsto & \varphi_{q,p}(t) = \alpha_p \varphi_q(t) \end{array}$$

By substituting those definitions we get the implication, explicitly, starting from theorem 2.1.2 we recover theorem 2.1.1 as

$$f(x_1, \ldots, x_n) = \sum_{k=1}^{2n+1} h\left(\sum_{i=1}^{n} \alpha_i \varphi_k(x_i)\right) = \sum_{k=1}^{2n+1} \Phi_k\left(\sum_{i=1}^{n} \varphi_{k,i}(x_i)\right)$$

□

In order to construct the proof we are going to use a chain of slightly easier lemmas and theorems that are equivalent to the above theorem. Every step is proved rigorously, to show the equivalences between slabs on the chain. First it is worth making a few definitions.

Definition 2.1.1. We can define the **ceiling function** (resp. **flooring function**) as the function that given a number returns the closer upper (resp. lower) integer to it. Mathematically

$$\begin{array}{rccl} \lceil\cdot\rceil: & \mathbb{R} & \to & \mathbb{Z} \\ & x & \mapsto & \lceil x \rceil = \min\limits_{z \in \mathbb{Z}: z \geq x} z \end{array} \quad \left(\text{resp.} \quad \begin{array}{rccl} \lfloor\cdot\rfloor: & \mathbb{R} & \to & \mathbb{Z} \\ & x & \mapsto & \lfloor x \rfloor = \max\limits_{z \in \mathbb{Z}: z \leq x} z \end{array}\right)$$

Observation. For the construction of the proof we set $\alpha_1, \ldots, \alpha_n$ to be square roots of pairwise different prime numbers, for simplicity we take a_i to be the square root of the i-th prime (starting by one).

Example. Let $n = 5$, then we have $\alpha_1 = 1$, $\alpha_2 = \sqrt{2}$, $\alpha_3 = \sqrt{3}$, $\alpha_4 = \sqrt{5}$, $\alpha_5 = \sqrt{7}$,

Definition 2.1.2. We define $\tilde{A}$ to be $\tilde{A} = \sum\limits_{i=1}^{n} \alpha_i$ and A to be $\lceil \tilde{A} \rceil$

Notation. We denote the closed interval $[0,1]$ as I

Notation. Throughout this section any reference to a function or map will refer to a continuous function or continuous map unless explicitly stated otherwise.

Definition 2.1.3. Given an ordered set of functions $\varphi_1, \dots, \varphi_n : I \to I$, we define φ to be a **map** or **vector-function** and it is defined as

$$\begin{array}{rccl} \varphi: & I & \to & I^n \\ & t & \mapsto & \varphi(t) = (\varphi_1(t), \varphi_2(t), \dots, \varphi_n(t)) \end{array}$$

Definition 2.1.4. Given a function $h : \mathbb{R} \to \mathbb{R}$ and a map φ we can define the function $S_\varphi h$ as

$$\begin{array}{rccl} S\varphi h: & I^n & \to & \mathbb{R} \\ & (x_1, \dots, x_n) & \mapsto & S_\varphi h(x_1, \dots, x_n) = \sum\limits_{k=1}^{2n+1} h\left(\sum_{i=1}^{n} \alpha_i \varphi_k(x_i)\right) \end{array}$$

Definition 2.1.5. For a compact subset $M \subset \mathbb{R}^{2n+1}$ and a function $f : M \to \mathbb{R}$, we define the norm of a function as

$$\|f\| = \sup_{z \in M} |f(z)| \tag{2.1}$$

Observation. $\|f\| = \|f\|_\infty$

Observation. Also since the set M is by definition compact, the value of $\|f\|$ is always attained.

Definition 2.1.6. A λ-prekolmogorov map for a non-zero function $f : I^n \to \mathbb{R}$ is a map $\varphi : I \to I^{2n+1}$ for which it exists a function $h : [0, A] \to R$ such that

$$\|f - S_\varphi h\| < \lambda \|f\| \text{ and } \|h\| \leq \|f\|$$

Observation. For $f = 0$ any map $\varphi : I \to I^{2n+1}$ is λ-prekolmogorov. It suffices with taking $h = 0$

Definition 2.1.7. Let E be a set and $(f_n)_{n \in \mathbb{N}} : E \to \mathbb{R}$ be a sequence of real-valued functions, we say the sequence is **uniformly convergent** on E with limit $f : E \to \mathbb{R}$ if for every $\varepsilon > 0$ exists a natural number N such that $\forall n \geq N$ and for all $x \in E$

$$|f_n(x) - f(x)| < \varepsilon$$

Theorem 2.1.3. *There is a map $\varphi : I \to I^{2n+1}$ which is $\frac{2n+3}{2n+4}$-prekolmogorov for any function $f : I^n \to \mathbb{R}$*

Proposition 2.1.2. *We can deduce theorem 2.1.2 from theorem 2.1.3*

Proof. We can define the functions $h : [0, A] \to \mathbb{R}$ via an iterative process constructing intermediate functions $h_m : [0, A] \to \mathbb{R}$ with m a positive integer. We start the process by defining $h_0 = 0$, and we construct h_m by applying theorem 2.1.3 to the function

$$f - \sum_{k=0}^{m-1} S_\varphi h_k$$

The map is $\frac{2n+3}{2n+4}$-prekolmogorov, and there is a function h_m with the properties from the λ-prekolmogorovity definition (2.1.6), that is, by setting $\lambda = \frac{2n+3}{2n+4}$, we have

$$\left\| f - \sum_{k=0}^{m} S_\varphi h_k \right\| < \lambda \left\| f - \sum_{k=0}^{m} S_\varphi h_k \right\| < \cdots < \lambda^m \|f\|$$

And also

$$\|h_m\| < \left\| f - \sum_{k=0}^{m-1} S_\varphi h_k \right\| < \lambda^{m-1} \|f\|$$

As $\lambda < 1$, we have that the functional series $\sum_{m=0}^{\infty} h_m$ uniformly converges to a function $h : [0, A] \to \mathbb{R}$, then we also have that

$$\sum_{m=0}^{\infty} S_\varphi h_m = \sum_{m=0}^{\infty} \sum_{k=1}^{2n+1} h_k \left(\sum_{i=1}^{n} \alpha_i \varphi_k(x_i) \right) = \sum_{k=1}^{2n+1} \sum_{m=0}^{\infty} h_k \left(\sum_{i=1}^{n} \alpha_i \varphi_k(x_i) \right) = S_\varphi \sum_{m=0}^{\infty} h_m = S_\varphi h$$

And also as $\lambda < 1$, $\lim_{m \to \infty} \lambda^m = 0$, taking the limit on the above inequality we get $\|f - S_\varphi h\| = 0$ or equivalently $S_\varphi h = f$, and by extending h to $\mathbb{R}$, we have

$$f = S_\varphi h = \sum_{k=1}^{2n+1} h \left(\sum_{i=1}^{n} \alpha_i \varphi_k(x_i) \right)$$

□

Lemma 2.1.1. *There is a map $\varphi : I \to I^{2n+1}$ which is $\frac{2n+2}{2n+3}$-prekolmogorov for any polynomial $g : I^n \to \mathbb{R}$ with rational coefficients.*

Theorem 2.1.4 (Heine-Cantor)**.** *If (X, d_X) and (Y, d_Y) are metric spaces, $f : X, Y$ is a continuous function and X is compact, then f is uniformly continuous*

Proof. See appendix A.1 □

Theorem 2.1.5 (Weierstrass rational approximation)**.** *For any continuous function $f : [0, 1] \to \mathbb{R}$ and any $\varepsilon > 0$, exists a polynomial $g \in \mathbb{Q}[x]$ (a polynomial of rational coefficients) such that $\forall x \in [0, 1]$ we have $|f(x) - g(x)| < \varepsilon$*

Proof. See appendix A.2 □

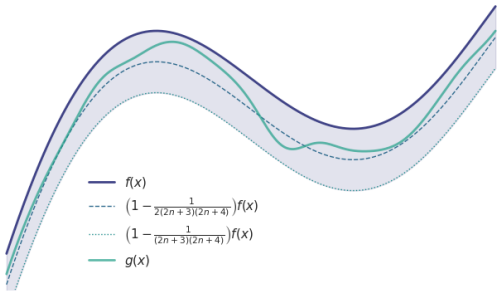


Figure 2.1: Setup for proof of theorem 2.1.3. By approximating the dashed function with a small enough tolerance, g never escapes the shaded area.

Observation. We can take the coefficients of q such that they satisfy that $\|q\| \leq \|f\|$

We can now prove theorem 2.1.3 (for the moment assuming true lemma 2.1.1)

Proof of theorem 2.1.3. Let $f : I^n \to \mathbb{R}$ be a function. Let $\varepsilon = 1/(2n+3)(2n+4)$. Then by the Weierstrass rational approximation theorem (2.1.5), we know that there is a polynomial $g : I^n \to \mathbb{R}$ with rational coefficients that approximates $(1-\varepsilon/2)f$ with greater precision than $\varepsilon/2$ (for a clearer picture of the setup, see the figure 2.1). This way we have a polynomial g that satisfies

$$\|g\| < \|f\| \text{ and } \|f-g\| < \frac{1}{\varepsilon}\|f\|$$

Now we can consider the map φ given by 2.1.1, as φ is $\frac{2n+2}{2n+3}$-prekolmogorov for g, there is a function $h : [0,A] \to \mathbb{R}$ such that

$$\|h\| \leq \|g\| < \|f\| \text{ and } \|g - S_\varphi h\| < \frac{2n+2}{2n+3}\|g\| < \frac{2n+2}{2n+3}\|f\|$$

Now by triangular inequality we get

$$\|f - S_\varphi h\| \leq \|f-g\| + \|g - S_\varphi h\| < \frac{1}{\varepsilon}\|f\| + \frac{2n+2}{2n+3}\|f\| = \frac{2n+3}{2n+4}\|f\|$$

So that we see that φ is $\frac{2n+3}{2n+4}$-prekolmogorov for f □

Lemma 2.1.2. *Given any function $f : [0,1]^n \to \mathbb{R}$, we have*

- *(stability) any map close enough to a $\frac{2n+2}{2n+3}$-prekolmogorov map for f is a $\frac{2n+2}{2n+3}$-prekolmogorov map for f*
- *(approximation) for any map $\varphi : I \to I^{2n+1}$ for f there is an arbitrarily close $\frac{2n+2}{2n+3}$-prekolmogorov map for f*

Notation. Given a function $f : I^n \to \mathbb{R}$, we denote by $PK_{\frac{2n+2}{2n+3}}(f) = PK(f)$ the set $\{\varphi : I \to I^{2n+1} \text{ s.t. } \varphi \text{ is } \frac{2n+2}{2n+3}\text{-prekolmogorov for } f\}$

Definition 2.1.8. A subset A of a topological space X is **dense** if A intersects every non-empty open subset of X

Definition 2.1.9. A topological space X is called a **Baire space** if every countable intersection of dense open sets is dense

Definition 2.1.10. A sequence $\{x_n\}_n \subset X$ of a metric space (X, d_X) is called **Cauchy** if $\forall \varepsilon \in \mathbb{R}_{>0}$, $\exists N \in \mathbb{N}$ such that $\forall n, m > N$, $d_X(x_m, x_n) < \varepsilon$

Definition 2.1.11. A metric space (X, d) is **complete** if every Cauchy sequence in X converges in X. I.e. the limit is also in X

Definition 2.1.12. A **pseudometric space** (X, d_X) is a set X together with a non-negative real-valued function $d_X : X \times X \to \mathbb{R}_{\geq 0}$ such that $\forall x, y, z \in X$

1. $d_X(x, x) = 0$
2. $d_X(x, y) = d_X(y, x)$
3. $d_X(x, z) \leq d_X(x, y) + d_X(y, z)$

Observation. Pseudometric spaces are a generalization of metric spaces, where points don't need to be distinguishable, in other words, it might be $d_X(x, y) = 0$ for points $x, y \in X$, $x \neq y$

Theorem 2.1.6 (Baire). *Every complete pseudometric space is a Baire space*

Proof. Let $\{U_j\}_{j \in J}$ be a countable collection of open dense subsets of a complete pseudometric space (X, d_X), we want to see that $\bigcap_{j \in J} U_j$ is dense. We have to show that any non-empty open subset $W \in X$ has some point $x \in W$ in common with every U_j.
As U_1 is a dense set, we know that $W \cap U_1 \neq \emptyset$, and since W and U_1 are open, we know that $\exists x_1 \in X$ and $0 < r_1 < 1$ such that $\bar{B}(x_1, r_1) \subset W \cap U_1$. We also know that the intersection $W \cap U_1$ is a non-empty open subset of X, and since U_2 is an open dense subset of X we know that $\exists x_2 \in (W \cap U_1) \cap U_2$ and $0 < r_2 < \frac{1}{2}$ such that $\bar{B}(x_2, r_2) \subset (W \cap U_1) \cap U_2$.
By iterating this argument, we end up with sequences $\{x_n\}_n$ and $\{r_n\}_n$ with $0 < r_i < \frac{1}{i}$ such that $\bar{B}(x_i, r_i) \subset B(x_{i-1}, r_{i-1}) \cap U_i$.
The sequence $\{x_n\}_n$ is Cauchy as $x_n \in B(x_m, r_m)$ if $n > m$ and by completeness of X, the sequence has a limit in $x \in X$.
By construction, we have that $\forall i$, $x \in \bar{B}(x_i, r_i)$, and hence $x \in W$ and $x \in U_j$ for all $j \in J$. With this we see that every countable intersection of dense open sets is dense, thus proving that the space X is a Baire space. □

Proposition 2.1.3. *We can deduce lemma 2.1.1 from 2.1.2*

Proof. Let $Q[x_1, \dots, x_n]$ be the set of all polynomials $I^n \to \mathbb{R}$ with rational coefficients. For any $g \in \mathbb{Q}[x_1, \dots, x_n]$, the set $PK_{\frac{2n+2}{2n+3}}(g)$ is open and dense in the space of all continuous maps $I \to I^{2n+1}$. We can better see this claim by observing that the openness is given by lemma 2.1.2-stability, as it tells us that there is an open neighbourhood around each of the points of $PK(g)$. Also we can see the density because lemma 2.1.2-approximation, which tells us that for any map $\varphi : I \to I^{2n+1}$, there is an arbitrarily close map in $PK(g)$.
As the space of all continuous maps $I \to I^{2n+1}$ with the supreme distance is complete, we can apply Baire theorem, and hence the countable intersection (due to the fact that the polynomials have rational coefficients and $\mathbb{Q}$ is countable)

$$\bigcap_{g \in \mathbb{Q}[x_1, \dots, x_n]} PK(g)$$

is dense and non-empty. Any map φ from this intersection is as required for lemma 2.1.1 □

Now we are in disposition of proving lemma 2.1.2. We will first prove the stability part, and later (with some more necessary setup) we will prove the approximation one.

Proof of lemma 2.1.2-stability. Let $\psi \in PK(f)$ be a map. Then $\exists h : [0, A] \to \mathbb{R}$ such that $\|f - S_\varphi h\| < \frac{2n+2}{2n+3}\|f\|$ we take the value

$$\varepsilon = \frac{1}{2n+1}\left(\frac{2n+2}{2n+3}\|f\| - \|f - S_\psi h\|\right)$$

we can see that $\varepsilon > 0$. By Heine-Cantor theorem, we have that the function h is uniformly continuous (theorem A.1.1), i.e. $\forall \varepsilon > 0$, $\exists \delta = \delta(\varepsilon) > 0$ such that if $|x - y| < \delta$, then $|h(x) - h(y)| < \varepsilon$
We want to see that any map φ such that $\|\varphi - \psi\| < \frac{\delta}{A}$, also belongs to $PK(f)$. For all $x, y \in I$ and $\forall k \in [1..2n+1]$, we have that

$$\left|\sum_{i=1}^{n} \alpha_i \varphi_k(x_i) - \sum_{i=1}^{n} \alpha_i \psi_k(x_i)\right| = \left|\sum_{i=1}^{n} \alpha_i \left(\varphi_k(x_i) - \psi_k(x_i)\right)\right| \leq \left|\sum_{i=1}^{n} \alpha_i\right| \frac{\delta}{A} < \delta$$

And because of that

$$\left|h\left(\sum_{i=1}^{n} \alpha_i \varphi_k(x_i)\right) - h\left(\sum_{i=1}^{n} \alpha_i \psi_k(x_i)\right)\right| < \varepsilon$$

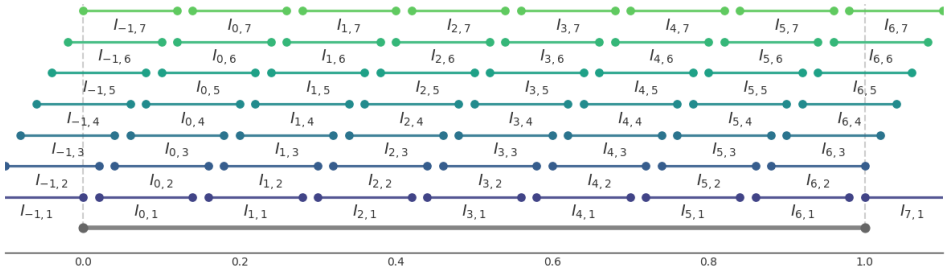


Figure 2.2: Intervals $I_{jk}(N)$ over the domain $[0,1]$ with $n = 3$ and $N = 50$

so by triangular inequality

$$\|S_\psi h - S_\varphi h\| = \left\| \sum_{k=1}^{2n+1} h\left(\sum_{i=1}^{n} \alpha_i \psi_k(x_i) \right) - \sum_{k=1}^{2n+1} h\left(\sum_{i=1}^{n} \alpha_i \varphi(x_i) \right) \right\| < (2n+1)\varepsilon$$

By applying triangular inequality again we have that

$$\|f - S_\varphi h\| \leq \|f - S_\psi h\| + \|S_\psi h - S_\varphi h\| < \frac{2n+2}{2n+3}\|f\| - (2n+1)\varepsilon + (2n+1)\varepsilon$$

Thus showing that $\|f - S_\varphi h\| < \frac{2n+2}{2n+3}\|f\|$, hence $\varphi \in PK(f)$ □

Notation. The expressions $[a,b] + c$ or $c + [a,b]$ denote the interval $[a + c, b + c]$, and the expressions $[a,b]c$ or $c[a,b]$ denote the interval $[ac, bc]$

Definition 2.1.13. For any $k \in [1..2n+1]$ we define

$$Z_k = Z_k(N) = \left[-1, \frac{N-k}{2n+1} \right] \cap \mathbb{Z}$$

Definition 2.1.14. For every $j \in Z_k(N)$, we define the interval

$$I_{j,k} = I_{j,k}(N) = \frac{2nI + (2n+1)j + k}{N} = \frac{2n[0,1] + (2n+1)j + k}{N}$$

Also we define the values

$$l_{j,k} = l_{j,k}(N) = \frac{(2n+1)j + k}{N} \quad \text{and} \quad l^0_{j,k} = \max\{l_{j,k}, 0\}$$

For a graphical representation of those intervals see the figure 2.2

Observation. For a given value of k, all intervals $I_{j,k}$ have length $\frac{2n}{N}$, and are disjoint by at least $\frac{1}{N}$

Observation. $l_{j,k}$ is the leftmost point of the interval $I_{j,k}$

Observation. For any value of $k \in [1..2n+1]$, $0 \in I_{-1,k}$ and $1 \in I_{\max Z_k,k}$

Proposition 2.1.4. *Let N be a positive integer, for any $(x_1,\dots,x_n) \in I^n$ and for at least $n+1$ values of $k \in [1..2n+1]$ there are $i_1,\dots,i_n \in Z_k(N)$ such that*

$$(x_1,\dots,x_n) \in I_{i_1,k}(N) \times \cdots \times I_{i_n,k}(N)$$

Proof. The fact that $(x_1,\dots,x_n) \in I_{i_1,k}(N) \times \cdots \times I_{i_n,k}(N)$ is equivalent to having $(Nx_1,\dots,Nx_n) \in I_{i_1,k}(1) \times \cdots \times I_{i_n,k}(1)$.
Restricting ourselves to the first coordinate, we consider the integer quotient $m = \left\lceil \frac{Nx_1}{2n+1} \right\rceil$ and also the remainder r of the same quotient. Then we know that $Nx_1 \in [(2n+1)m + r, (2n+1)m + r + 1]$, so for any $s = 0, 1, \dots, 2n-1$, we have

$$Nx_1 \in [(2n+1)m + r - s, (2n+1) + r - s + 2n] = 2nI + (2n+1)m + (r-s)$$

Hence, considering $k = r - s(\mod 2n+1)$, there is $j \in Z_k(N)$ such that $Nx_1 \in I_{j,k}(1)$, then for at least $2n$ values of k we have an appropriate value of j.
We can repeat the same argument for all other dimensions, obtaining $2n$ values of j for each. We then have a minimum overlap of $n+1$ values of k (as we have $2n$ elements n times of $2n+1$ possible ones). This tells us that there are at least $n+1$ distinct tuples $(i_1,\dots,i_n)$ with $i_i \in Z_k(1)$ such that $(Nx_1,\dots,Nx_n) \in I_{i_1,k}(1) \times \cdots \times I_{i_n,k}(1)$ as we wanted to see. □

Observation. The above proposition proves that for any given point in $[0,1]^n$ there are at least $n+1$ hypercubes $I_{i_1,k} \times \cdots \times I_{i_n,k}$ containing it.

Definition 2.1.15. A function $\varphi : I \to I$ **rationally separates** a number of pairwise disjoint closed intervals on the line if φ has constant rational pairwise different values on such intervals

Notation. For a map $\varphi : I \to I^{2n+1}$, we denote $\|\varphi\| = \max\limits_{k \in [0..2n+1]} \|\varphi_k\|$

Proposition 2.1.5. *Given $\varepsilon > 0$ and a function $\psi : I \to I$. Letting $k \in [0..2n+1]$, there is an integer $N_0 = N_0(\psi, k) > 0$ such that $\forall N > N_0$ and for any finite set G of rational numbers, there is a function $\varphi : I \to I$ such that*

- $\|\varphi - \psi\| < \varepsilon$
- *φ rationally separates the family of intervals $I_{j,k}(N)$, where $j \in Z_k(N)$*
- *$\varphi(I_{j,k}) \notin G$ for any $j \in Z_k(N)$*

Proof. As ψ is continuous in I, it is uniformly continuous (by Heine-Cantor theorem (A.1.1)), so $\exists \delta = \delta(\varepsilon) > 0$ such that

$$|\psi(x) - \psi(y)| < \varepsilon \text{ if } |x - y| < \delta$$

We now take $N_0 = \left\lceil \frac{2n+1}{\delta} \right\rceil$. Now for any $N > N_0$, we define a piecewise linear function $\varphi : I \to I$ such that:

- For each interval $I_{j,k}$, the value $\varphi(I_{j,k})$ is a constant rational and is satisfies

  $$\left| \varphi(x) - \psi(l^0_{j,k}) \right| < \frac{\varepsilon}{2}$$

  for all points in $I_{j,k}$
- All values $\varphi(I_{j,k})$ are pairwise different and are not in G
- On the gaps between $I_{j,k}$ and $I_{j+1,k}$, the function φ is linear

As the points $\{0, 1\}$ are covered by the intervals $I_{-1,k}$ and $I_{\max Z_k,k}$ respectively, the function is well defined at those points. By construction, φ rationally separates the family of intervals $\{I_{j,k}\}$ and its values are different from those at G. The only remaining thing to prove is that

$$|\varphi(x) - \psi(x)| < \varepsilon \text{ for all } x \in [0, 1]$$

We are proving this inequality by cases, first let us consider $x \in I_{j,k}$ for some j and k, then we have $\varphi(x) = \varphi(l^0_{j,k}$, and since the interval $I_{j,k}$ has length $\frac{2n}{N}$, we have that $|x - l^0_{j,k}| \leq \frac{2n}{N} < \frac{2n+1}{N} = \delta$, which allows us to write by triangular inequality

$$|\varphi(x) - \psi(x)| \leq |\varphi(l^0_{j,k}) - \psi(l^0_{j,k})| + |\psi(l^0_{j,k}) - \psi(x)| < \frac{\varepsilon}{2} + \frac{\varepsilon}{2} = \varepsilon$$

Now considering a point x such that $x \notin I_{j,k}$, we can still find a value $j \in Z_k(N)$ such that x lies on the empty space between $I_{j,k}$ and $I_{j+1,k}$. Alternatively

$$l_{j,k} + \frac{2n}{N} < x < l_{j+1,k} = l^0_{j+1,k}$$

since we also have the relations between ends of intervals $l_{j,k} + \frac{2n+1}{N} = l_{j+1,k}$, we know that there is a value $\alpha \in (0, 1)$ such that $x = l_{j+1,k} - \frac{\alpha}{N}$, also, as φ is linear in the gap $l_{j+1,k} + [-\frac{1}{N}, 0]$, we have that

$$\varphi(x)|_{[l_{j+1,k} - \frac{1}{N}, l_{j+1,k}]} = \alpha\varphi\left(l_{j+1,k} - \frac{1}{N}\right) + (1 - \alpha)\varphi\left(l_{j+1,k}\right)$$

And knowing that we can write $\psi(x) = \alpha\psi(x) + (1-\alpha)\psi(x)$ we obtain the inequality

$$\begin{aligned}|\varphi(x) - \psi(x)| &= \left|\alpha\varphi\left(l_{j+1,k} - \frac{1}{N}\right) + (1-\alpha)\varphi\left(l_{j+1,k}\right) - (\alpha\psi(x) + (1-\alpha)\psi(x))\right| \\ &= \left|\alpha\left[\varphi\left(l_{j+1,k} - \frac{1}{N}\right) - \psi(x)\right] + (1-\alpha)\left[\varphi(l_{j+1,k}) - \psi(x)\right]\right| \qquad (2.2) \\ &\leq \alpha\left|\varphi\left(l_{j+1,k} - \frac{1}{N}\right) - \psi(x)\right| + (1-\alpha)\left|\varphi(l_{j+1,k}) - \psi(x)\right|\end{aligned}$$

Considering that $\varphi(l_{j+1,k} - \frac{1}{N}) = \varphi(l^0_{j,k})$ and that $|x - l^0_{j,k}| \leq \frac{2n+1}{N} < \delta$, we can bound the first term as

$$\left|\varphi\left(l_{j+1,k} - \frac{1}{N}\right) - \psi(x)\right| \leq |\varphi(l^0_{j,k}) - \psi(l^0_{j,k})| + |\psi(l^0_{j,k}) - \psi(x)| < \frac{\varepsilon}{2} + \frac{\varepsilon}{2} = \varepsilon$$

and the second term can be bounded observing that $|l_{j+1,k} - x| < \frac{1}{N} < \delta$, hence

$$|\varphi(l_{j+1,k}) - \psi(x)| \leq |\varphi(l_{j+1,k}) - \psi(l_{j+1,k})| + |\psi(l_{j+1,k}) - \psi(x)| < \frac{\varepsilon}{2} + \frac{\varepsilon}{2} = \varepsilon$$

Hence substituting in inequality 2.2, we have that

$$|\varphi(x) - \psi(x)| < \alpha\varepsilon + (1-\alpha)\varepsilon = \varepsilon$$

As we wanted to show □

Corollary 2.1.5.1. *Let $\varepsilon > 0$ be a positive number, and $\psi : I \to I^{2n+1}$ be a map, then there is an integer $N_0 > 0$ such that $\forall N \geq N_0$ integer, there is a map $\varphi : I \to I^{2n+1}$ satisfying*

- $\|\varphi - \psi\| < \varepsilon$
- *For every $k \in [1..2n+1]$, the function φ_k rational separates the family of intervals $\{I_{j,k}\}$ provided by $j \in Z_k(N)$*
- *Numbers $\varphi_k(I_{j,k})$ are pairwise different for different $k \in [1..2n+1]$ and $j \in Z_k(N)$*

Proof. We can choose $N_0 = \max_{k\in[1..2n+1]} N_0^k$, where $N_0^k = N_0(\psi_k, k)$ provided by proposition 2.1.5. Consider now $N > N_0$, by applying proposition 2.1.5 to ψ_1, $k = 1$ and $G = \emptyset$, this yields a function φ_1 satisfying the hypothesis for ψ_1.

Now we can apply proposition 2.1.5 again to ψ_2, $k = 2$ and $G = \{\varphi_1(I_{j,1})\}_{j\in Z_1(N)}$, this gives us a function φ_2. By repeating this process, on step s we apply proposition 2.1.5 to ψ_s, $k = s$ and

$$G = \bigcup_{r=1}^{s-1} \{\varphi_r(I_{j,r})\}_{j\in Z_r(N)}$$

After repeating this process for $2n + 1$ times we can construct the map $\varphi = (\varphi_1, \ldots, \varphi_{2n+1})$, that proves the theorem □

Proof of lemma 2.1.2-approximation. Fixed a map $\psi : I \to I^{2n+1}$ and $\varepsilon > 0$, we want to see that there is $\varphi \in PK_{\frac{2n+2}{2n+3}}(f)$ that is ε-close to ψ.
By applying corollary 2.1.5.1 to function ψ and ε, we obtain a number N_0. Since f is uniformly continuous (by Heine-Cantor theorem (A.1.1)), we can find a positive integer $N > N_0$ such that

$$|f(x_1, \ldots, x_n) - f(x'_1, \ldots, x'_n)| < \frac{1}{2n+2}\|f\| \text{ if } |x_i - x'_i| < \frac{2n}{N} \text{ for all } i \in [1..n]$$

Now by the corollary 2.1.5.1, and using this value of N, we can construct the map φ. For each $k \in [1..2n + 1]$ we can now define the function

$$\begin{array}{rccl} \tilde{\varphi}_k : & I^n & \to & [0, A] \\ & (x_1, \ldots, x_n) & \mapsto & \tilde{\varphi}_k(x_1, \ldots, x_n) = \sum_{i=1}^{n} \alpha_i \varphi_k(x_i) \end{array}$$

We can show that $\tilde{\varphi}_k$ has different constant values on the hypercubes that are cartesian products of intervals $I_{j,k}$ by considering $x_1, \ldots, x_n, x'_1, \ldots, x'_n \in I_{j,k}$ such that

$$\sum_{i=1}^{n} \alpha_i \varphi_k(x_i) = \sum_{i=1}^{n} \alpha_i \varphi_k(x'_i)$$

By definition, $\varphi_k(x_1), \ldots, \varphi_k(x_n), \varphi_k(x'_1), \ldots, \varphi_k(x'_n)$ are rational, and since α_i are roots of pairwise different prime numbers we have that necessarily $\varphi_k(x_i) = \varphi_k(x'_i)$, for all $i \in [0..n]$. This means that points $\{x_i, x'_i\}$ lie in the same interval, and therefore, tuples $(x_1, \ldots, x_n)$ and $(x'_1, \ldots, x'_n)$ both lie in the same hypercube.
Now we want to see that the value $\tilde{\varphi}_k(I_{i_1,k} \times \cdots \times I_{i_n,k})$ is pairwise different for pairwise different tuples $(i_1, \ldots, i_n, k)$. Indeed, we can suppose $\tilde{\varphi}_k(I_{i_1,k} \times \cdots \times I_{i_n,k}) = \tilde{\varphi}_n(I_{j_1,n} \times \cdots \times I_{j_n,n})$ and show that necessarily $i_i = j_i$, $\forall i$ and $k = n$. Restricting to each of the dimensions at a time, we have equalities such as $\varphi_k(I_{i_i,k}) = \varphi_n(I_{j_i,n})$ (again we've considered that α_i are square roots of pairwise distinct primes, and since φ_k is rational, we need the equality to hold). This way we have that $k = n$ and $i_i = j_i$, repeating this argument for all possible values of i we observe that different tuples yield different evaluations.

To continue with the proof we now define a piecewise linear function $h : [0, A] \to \mathbb{R}$ such that $h\left(\tilde{\varphi}_k(I_{i_1,k} \times \cdots \times I_{i_n,k})\right) = \frac{1}{n+1} f(l^0_{i_1,k}, \cdots l^0_{i_n,k})$ for any $k \in [1..2n+1]$ and $i_i \in Z_k(N)$. This definition guarantees that $\|h\| \leq \frac{1}{n+1}\|f\|$ (as h is linear and bounded to be $\frac{1}{n+1}$ of a specific value of f which might or might not be maximum). As f is uniformly continuous, we have $|f(z) - f(z')| < \frac{1}{2n+2}\|f\|$ for any $k \in [1..2n+1]$, for all $i_i \in Z_k(N)$ and any $z, z' \in I_{i_1,k} \times \cdots \times I_{i_n,k}$. Then by proposition 2.1.4 there are $n+1$ tuples $(i_{1,s}, \ldots i_{n,s}, k_s) \in \underbrace{Z_{k_s}(N) \times \cdots Z_{k_s}(N)}_{n \text{ times}} \times [1..2n+1] = Z^n_{k_s} \times [1..2n+1]$, indexed by $s \in [1..n+1]$ that contain point z, i.e. $z \in I_{i_{1,s}} \times \cdots \times I_{i_{n,s}}$. For those

$$h(\tilde{\varphi}_{k_s}(z)) = \frac{1}{n+1} f(l^0_{i_{1,s},k_s}, \ldots, l^0_{i_{n,s},k_s})$$

So we have

$$\left| h\left(\tilde{\varphi}_{k_s}(z)\right) - \frac{1}{n+1} f(z) \right| = \left| \frac{1}{n+1} f(l^0_{i_{1,s},k_s}, \ldots, l^0_{i_{n,s},k_s}) - \frac{1}{n+1} f(z) \right|$$

$$= \frac{1}{n+1} \left| f(l^0_{i_{1,s},k_s}, \ldots, l^0_{i_{n,s},k_s}) - f(z) \right| \leq \frac{1}{n+1} \frac{1}{2n+2} \|f\| = \frac{1}{2(n+1)^2} \|f\|$$

And then, when considering all values of k_s, we have (by triangular inequality)

$$\left| \sum_{s=1}^{n+1} h\left(\tilde{\varphi}_{k_s}(z)\right) - f(z) \right| \leq \overbrace{\left| h(\tilde{\varphi}_{k_1}(z)) - \frac{1}{n+1} f(z) \right| + \cdots + \left| h(\tilde{\varphi}_{k_{n+1}}(z)) - \frac{1}{n+1} f(z) \right|}^{n+1 \text{ times}}$$

$$\leq \overbrace{\frac{1}{2(n+1)^2}\|f\| + \cdots + \frac{1}{2(n+1)^2}\|f\|}^{n+1 \text{ times}} = \frac{1}{2(n+1)}\|f\|$$

And finally, we can consider all possible indices of k as $\{k_s\}_{s \in [1..n+1]} \cup \{k_r\}_{r \in [1..n]}$, where k_r are the indices that weren't selected previously according to the presence of z in the hypercube

$$\left| S_\varphi h(z) - f(z) \right| \leq \left| \sum_{s=1}^{n+1} h(\tilde{\varphi}_{k_s}(z)) + \sum_{r=1}^{n} h(\tilde{\varphi}_{k_r}(z)) - f(z) \right|$$

$$\leq \left| \sum_{s=1}^{n+1} h(\tilde{\varphi}_{k_s}(z)) - f(z) \right| + \left| \sum_{r=1}^{n} h(\tilde{\varphi}_{k_r}(z)) \right|$$

$$\leq \frac{1}{2(n+1)}\|f\| + \sum_{r=1}^{n} \frac{1}{n+1}\|f\| = \left(\frac{1}{2(n+1)} + \frac{n}{n+1} \right) \|f\| = \frac{2n+1}{2n+2}\|f\| < \frac{2n+2}{2n+3}$$

So we have seen that φ is indeed $\frac{2n+2}{2n+3}$-prekolmogorov
This concludes the proof of the Arnold-Kolmogorov representation theorem (2.1.1)
□

Chapter 3

Neural networks

3.1 Neural networks

3.1.1 Objective

Given pairs of points (or vectors) (x_i, y_i), the goal of a neural network is to find a function $f(x)$ that is capable of predicting the output value y for any x (specially when the datapoint has never been seen by the network).
This problem is not new, as predictions (mainly astronomical) are some of the earliest mathematical recordings in history. Classical approaches imposed a given mathematical framework to the datapoints, possibly missing some deeper structure on the data or simplifying a bit too much natural phenomena. Neural networks reinvent the way we tackle prediction tasks, as part of the process is converging to the best describing mathematical framework, this is all attained via flexible parametrization. Not everything is a bed of roses, as in order to have the flexibility to model virtually anything, the number of parameters typically range between a few million to over a trillion of parameters. Practically, this translates as thousands of hours of high performance computing.
Neural networks' structure aims to be similar to brain tissue, where neurons are interconnected to each other, and the network is capable of learning by strengthening or weakening such connections.
This all will be covered in detail in further subsections.

3.1.2 Building blocks

A neural network is structured in **layers**, and each of those layers consists on **neurons** or **nodes** that are interconnected via **weights** or **edges**. We can distinguish between three types of layers: an input layer, which has the same amount of nodes as of features (or dimensions the dataset has), hidden layers, there can

be any number of those and each can be arbitrarily big (even though typically the number of neurons decreases towards the final layers of the network), and an output layer, which is responsible of providing the prediction, the number of nodes in this layer depends on the amount of dimensions we want our prediction to have. The amount of layers is known as the **depth** of the network, and the number of neurons of each layer as the **width**. Wider and deeper networks can generalize better, but are more expensive to train and harder to interpret.

The weights of a neural network determine the importance that a given neuron j in layer $i-1$ has when passed to neuron k in layer i (morally, the strength of the connection between the two neurons). A neural network that is fully connected forward-feeding (that is all neurons in layer $i-1$ are connected to all neurons in layer i) is known as a MultiLayer Perceptron (MLP). MLPs were originally the building blocks of artificial intelligence models due to their capacity of generalization and their mathematical simplicity.

If necessary, in a MLP, we can get rid of some of those connections by just setting the corresponding weights to 0. Additionally to the weights we also find **biases** which are constant terms added to the neuron (we can think of those as independent terms that get summed to each weight).

We can describe this architecture with the help of mathematics, denoting by N_i the amount of neurons in layer i, and by $\omega^i_{j,k}$ the weight that connects neuron j in layer i to neuron k in layer $i+1$, then we can express the neuron k in layer $i+1$ (n^{i+1}_k) as

$$n^{i+1}_k(n^i_1,\dots,n^i_{N_i}) = \left(\sum_{j=1}^{N_i} \omega^i_{j,k} n^i_j\right) + b^i_k$$

where b^i_k denotes the bias that is applied to the neuron k in layer $i+1$.

This is perhaps the simplest of all possible neurons, where previous neurons are combined linearly to output the value of the newer neuron. This comes with one very restrictive limitation, a combination of linear maps can only yield a linear map. Hence, this kind of architecture is only useful for datapoints that are expressed by a linear behaviour, and this is a rather complicated way of obtaining the best-fit straight line.

However, the solution to this problem comes when adding a non-linearity within the neurons, so that the network is able to learn complicated behaviour, becoming useful for some other complex tasks. As its name indicates, a non-linearity is a non-linear function, in our context, when it is applied to a neuron, it modifies its output overcoming the limitation imposed by linear combination. Some of the most widely used non-linearities are ReLU (rectified linear unit), tanh or sigmoids, those functions typically have good derivative properties (it will later

become evident why)

$$\text{ReLu}(x) = \begin{cases} x & \text{if} \quad x \geq 0 \\ 0 & \text{if} \quad x < 0 \end{cases} = max(x, 0) \quad , \quad \text{sigmoid}(x) = \frac{1}{1+e^{-x}}$$

3.1.3 Forward pass

Denoting the non-linearity by σ_i and using vector notation $\boldsymbol{n}_i = (n_1^i, n_2^i, \dots, n_{N_i}^i)^T$, $\boldsymbol{\omega}_k^i = (\omega_{1,k}^i, \omega_{2,k}^i, \dots, \omega_{N_i,k}^i)^T$, we can express the neuron as a function

$$n_k^{i+1}(\boldsymbol{n}_i) = \sigma_i \left((\boldsymbol{\omega}_k^i)^T \boldsymbol{n}_i + b_k^i \right)$$

By looking at the expression of a neuron, one can see that a layer allows for a matrix expression, where by respectively defining the weight matrix and bias vector

$$\boldsymbol{\Omega}_i = \begin{pmatrix} \omega_{1,1}^i & \omega_{2,1}^i & \cdots & \omega_{N_i,1}^i \\ \omega_{1,2}^i & \omega_{2,2}^i & \cdots & \omega_{N_i,2}^i \\ \vdots & \vdots & & \vdots \\ \omega_{1,N_{i+1}}^i & \omega_{2,N_{i+1}}^i & \cdots & \omega_{N_i,N_{i+1}}^i \end{pmatrix} \quad \text{and} \quad \boldsymbol{b}_i = \begin{pmatrix} b_1^i \\ b_2^i \\ \vdots \\ b_{N_i+1}^i \end{pmatrix}$$

and assuming that there are L layers in total, we can express the i-th layer as the neuron vector

$$\boldsymbol{n}_{i+1} = \sigma_i(\boldsymbol{\Omega}_i \boldsymbol{n}_i + \boldsymbol{b}_i), \ 1 \leq i \leq L-1$$

The input layer ($\boldsymbol{n}_1$) doesn't have a non-linearity (or we can take it to be equal to the identity), as otherwise the fed data to the model would be altered from the very beginning. Also, in here we have considered that in a layer all neurons share the same non-linearity for simplicity, even though this could not be the case. Now, by composing those layer expressions, we can get the complete MLP description (considering L layers), we have

$$\hat{\boldsymbol{y}} = \text{MLP}(\boldsymbol{x}) = \boldsymbol{n}_L \circ \cdots \circ \boldsymbol{n}_1 = \sigma_{L-1} \left(\boldsymbol{\Omega}_{L-1} \left(\cdots (\boldsymbol{\Omega}_1 \boldsymbol{x} + b_1) \cdots \right) + \boldsymbol{b}_{L-1} \right)$$

Often, in the output layer a different non-linearity is used depending on the output we desire. For example for yes/ no decisions, a sigmoid is often used as it bounds the output between 0 and 1, making it suitable for that kind of tasks. For other uses such as embeddings, $tanh(x)$ is also a common choice as it bounds the data between -1 and 1.

In this section and further ones, we have considered the biases as sums in the expression to emphasize their independence to the datapoints, this could all be integrated in a larger matrix $(N_{i+1}) \times (N_i + 1)$, where the bias is set as an additional column. This new matrix $\boldsymbol{W}_i$ (corresponding to the affine map between one layer and another) would allow the multilayer perceptron to be written as

$$\hat{\boldsymbol{y}} = \text{MLP}(\boldsymbol{x}) = (\sigma_{L-1} \circ \boldsymbol{W}_{L-1} \circ \sigma_{L-1} \circ \boldsymbol{W}_{L-2} \circ \cdots \circ \sigma_2 \circ \boldsymbol{W}_2 \circ \sigma_1 \circ \boldsymbol{W}_1)\boldsymbol{x}$$

3.1.4 Evaluation and training

So far we have defined a forward step on the neural network, but the model has not learned anything from the data, so this section aims to understand the procedure that the network follows in order to tune its parameters to obtain better predictions/ results.

The first method that we need is a measure of how well the current predictions are with respect to the real values. This measure is known as the **loss function** $\mathcal{L}(\boldsymbol{y}, \hat{\boldsymbol{y}})$. The image of this function is a measure of how close are the values $\hat{\boldsymbol{y}}$ (the output of our neural network) with the values $\boldsymbol{y}$ the real ones coming from the training set. Some common choices of loss function are the mean squared error (MSE), or entropy-related measures (typically variations of Shannon entropy). The only information that the neural network has about its performance is the output of $\mathcal{L}$, and it trains to minimize the value of it. Most times what keeps a model from performing well is the pick of the loss function.

Minimizing the value of $\mathcal{L}$ without further consideration is not enough, as doing this generates a model that performs excellently with our dataset, but fails hideously with any non-seen datapoint; this situation is known as **overfitting**. In order to avoid this, commonly the dataset is split into three pieces, which are denoted **training set**, **test set** and **validation set** with ratios around 0.7, 0.15 and 0.15 of the total number of datapoints respectively. This splitting is done in ways that each portion is representative of the elements present in the dataset. The model trains according to only the points in the training set, while in the evaluation of the loss function we use the points from the validation set, and we use those to also modify the hyperparameters in order to achieve a better result. This simple trick allows the model to generalize better. Finally the test set, is used as an unbiased performance indicator of the model; the test set is never used during the development of the model.

The way the learnable parameters in the network are tweaked is according to a technique called backpropagation, which morally consists on understanding which (and how) weights and biases should be changed to obtain a smaller value of the loss function.

Before going into the algorithm, it is worth computing the following partial derivatives

$$\frac{\partial(\boldsymbol{\Omega}_i \boldsymbol{n}_i + \boldsymbol{b}_i)}{\partial \boldsymbol{n}_i} = \frac{\partial}{\partial \boldsymbol{n}_i} \left[\begin{pmatrix} \omega^i_{1,1} & \cdots & \omega_{N_i,1} \\ \vdots & & \vdots \\ \omega_{1,N_{i+1}} & \cdots & \omega^i_{N_i,N_{i+1}} \end{pmatrix} \begin{pmatrix} n^i_1 \\ \vdots \\ n^i_{N_i} \end{pmatrix} + \begin{pmatrix} b^i_1 \\ \vdots \\ b^i_{N_{i+1}} \end{pmatrix} \right]$$

$$= \frac{\partial}{\partial \boldsymbol{n}_i} \begin{pmatrix} \sum_{j=1}^{N_i} \omega_{j,1}^i n_j^i + b_1^i \\ \vdots \\ \sum_{j=1}^{N_i} \omega_{j,N_{i+1}}^i n_j^i + b_{N_{i+1}}^i \end{pmatrix}$$

To ease the notation we can denote

$$S_k^i = \sum_{j=1}^{N_i} \omega_{j,k}^i n_j^i + b_k^i$$

and then we have

$$\frac{\partial(\boldsymbol{\Omega}_i \boldsymbol{n}_i + \boldsymbol{b}_i)}{\partial \boldsymbol{n}_i} = \frac{\partial}{\partial \boldsymbol{n}_i} \begin{pmatrix} S_1^i \\ \vdots \\ S_{N_{i+1}}^i \end{pmatrix} = \begin{pmatrix} \frac{\partial S_1^i}{\partial n_1^i} & \cdots & \frac{\partial S_1^i}{\partial n_{N_i}^i} \\ \vdots & & \vdots \\ \frac{\partial S_{N_{i+1}}^i}{\partial n_1^i} & \cdots & \frac{\partial S_{N_{i+1}}^i}{\partial n_{N_i}^i} \end{pmatrix} = \begin{pmatrix} \omega_{1,1}^i & \cdots & \omega_{N_i,1}^i \\ \vdots & & \vdots \\ \omega_{1,N_{i+1}}^i & \cdots & \omega_{N_i,N_{i+1}}^i \end{pmatrix} = \boldsymbol{\Omega}_i$$

Also, we compute the derivative

$$\frac{\partial(\boldsymbol{\Omega}_i \boldsymbol{n}_i + \boldsymbol{b}_i)}{\partial \boldsymbol{\Omega}_i} = \frac{\partial}{\partial \boldsymbol{\Omega}_i} \begin{pmatrix} S_1^i \\ \vdots \\ S_{N_{i+1}}^i \end{pmatrix}$$

which gives place to a 3-order tensor, whose *jkl*-component is

$$\frac{\partial S_j^i}{\partial \omega_{l,k}^i} = \frac{\partial}{\partial \omega_{l,k}^i} \sum_{r=1}^{N_i} \omega_{r,j}^i n_r^i + b_r^i = \sum_{r=1}^{N_i} \frac{\partial}{\partial \omega_{l,k}^i} \omega_{r,j}^i n_r^i = n_r^i \delta_{l,r} \delta_{k,j} = n_l^i$$

Note that the indices on the derivative seem to be permuted, but it has to do with the way $\boldsymbol{\Omega}$ is constructed. The above derivation shows that we can express

$$\frac{\partial(\boldsymbol{\Omega}_i \boldsymbol{n}_i + \boldsymbol{b}_i)}{\partial \boldsymbol{\Omega}_i} = \begin{pmatrix} n_1^i \\ \vdots \\ n_{N_i}^i \end{pmatrix} = \boldsymbol{n}_i$$

And lastly we have

$$\frac{\partial(\boldsymbol{\Omega}_i \boldsymbol{n}_i + \boldsymbol{b}_i)}{\partial \boldsymbol{b}_i} = \frac{\partial}{\partial \boldsymbol{b}_i} \begin{pmatrix} S_1^i \\ \vdots \\ S_{N_{i+1}}^i \end{pmatrix} = \begin{pmatrix} \frac{\partial S_1^i}{\partial b_1^i} & \cdots & \frac{\partial S_1^i}{\partial b_{N_{i+1}}^i} \\ \vdots & \ddots & \vdots \\ \frac{\partial S_{N_{i+1}}^i}{\partial b_1^i} & \cdots & \frac{\partial S_{N_{i+1}}^i}{\partial b_{N_{i+1}}^i} \end{pmatrix} = \begin{pmatrix} 1 & 0 & \cdots & 0 \\ 0 & 1 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & 1 \end{pmatrix} = \mathbb{I}_{N_{i+1}}$$

where $\mathbb{I}_{N_{i+1}}$ denotes the identity matrix $N_{i+1} \times N_{i+1}$.
Denoting $\boldsymbol{z}_i = \boldsymbol{\Omega}_i \boldsymbol{n}_i + \boldsymbol{b}_i$, the value of the layer previously to the application of the non-linearity, using the chain rule, we can determine that for a given layer $\boldsymbol{l}_i$ we have

$$\frac{\partial \boldsymbol{n}_{i+1}}{\partial \boldsymbol{n}_i} = \frac{\mathrm{d}\sigma_i(\boldsymbol{z}_i)}{\mathrm{d}\boldsymbol{z}_i}\frac{\partial \boldsymbol{z}_i}{\partial \boldsymbol{n}_i} = \frac{\mathrm{d}\sigma_i(\boldsymbol{z}_i)}{\mathrm{d}\boldsymbol{z_i}}\boldsymbol{\Omega}_i \qquad \frac{\partial \boldsymbol{n}_{i+1}}{\partial \boldsymbol{\Omega}_i} = \frac{\mathrm{d}\sigma_i(\boldsymbol{z}_i)}{\mathrm{d}\boldsymbol{z}_i}\frac{\partial \boldsymbol{z}_i}{\partial \boldsymbol{\Omega}_i} = \frac{\mathrm{d}\sigma_i(\boldsymbol{z}_i)}{\mathrm{d}\boldsymbol{z_i}}\boldsymbol{n}_i$$

$$\frac{\partial \boldsymbol{n}_{i+1}}{\partial \boldsymbol{b}_i} = \frac{\mathrm{d}\sigma_i(\boldsymbol{z}_i)}{\mathrm{d}\boldsymbol{z}_i}\frac{\partial \boldsymbol{z}_i}{\partial \boldsymbol{b}_i} = \frac{\mathrm{d}\sigma_i(\boldsymbol{z}_i)}{\mathrm{d}\boldsymbol{z_i}}\mathbb{I}_{N_{i+1}}$$

Now, we can understand how the loss function $\mathcal{L}$ is affected by any weights or bias by computing the corresponding partial derivative, which we can achieve by applying the chain rule. To ease further calculations, we can compute how the loss changes with respect to the pre-activated values $\boldsymbol{z}_i$, and denote it $\boldsymbol{\delta}_i$

$$\begin{aligned}\boldsymbol{\delta}_i = \frac{\partial \mathcal{L}}{\partial \boldsymbol{z}_i} &= \frac{\partial \mathcal{L}}{\partial \hat{\boldsymbol{y}}}\frac{\partial \hat{\boldsymbol{y}}}{\partial \boldsymbol{z}_i} = \frac{\partial \mathcal{L}}{\partial \hat{\boldsymbol{y}}}\frac{\partial \hat{\boldsymbol{y}}}{\partial \boldsymbol{n}_L}\frac{\partial \boldsymbol{n}_L}{\partial \boldsymbol{n}_{L-1}}\cdots\frac{\partial \boldsymbol{n}_{i+2}}{\partial \boldsymbol{n}_{i+1}}\frac{\partial \boldsymbol{n}_{i+1}}{\partial \boldsymbol{z}_i} \\ &= \frac{\partial \mathcal{L}}{\partial \hat{\boldsymbol{y}}}\frac{\partial \hat{\boldsymbol{y}}}{\partial \sigma_{L-1}(\boldsymbol{z}_{L-1})}\frac{\partial \sigma_{L-1}(\boldsymbol{z}_{L-1})}{\partial \boldsymbol{z}_{L-1}}\frac{\partial \boldsymbol{z}_{L-1}}{\partial \sigma_{L-2}(\boldsymbol{z}_{L-2})}\cdots\frac{\partial \sigma_i(\boldsymbol{z}_i)}{\partial \boldsymbol{z}_i} \\ &= \frac{\partial \mathcal{L}}{\partial \hat{\boldsymbol{y}}}\boldsymbol{\Omega}_L\frac{\partial \sigma_{L-1}}{\partial \boldsymbol{z}_{L-1}}\boldsymbol{\Omega}_{L-1}\cdots\frac{\partial \sigma_{i+1}}{\partial \boldsymbol{z}_{i+1}}\boldsymbol{\Omega}_{i+1}\frac{\partial \sigma_i}{\partial \boldsymbol{z}_i}\end{aligned}$$

We can observe that

$$\boldsymbol{\delta}_i = \boldsymbol{\delta}_{i+1}\frac{\partial \boldsymbol{z}_{i+1}}{\partial \boldsymbol{n}_i}\frac{\partial \boldsymbol{n}_i}{\partial \boldsymbol{z}_i} = \boldsymbol{\delta}_{i+1}\boldsymbol{\Omega}_{i+1}\frac{\partial \sigma_i}{\partial \boldsymbol{z}_i}$$

Now we can determine the effect that any bias or weight has with respect to the loss by further applying the chain rule, indeed

$$\frac{\partial \mathcal{L}}{\partial \boldsymbol{b}_i} = \frac{\partial \mathcal{L}}{\partial \boldsymbol{z}_i}\frac{\partial \boldsymbol{z}_i}{\partial \boldsymbol{b}_i} = \boldsymbol{\delta}_i\mathbb{I}_{N_i} = \boldsymbol{\delta}_i \qquad \frac{\partial \mathcal{L}}{\partial \boldsymbol{\Omega}_i} = \frac{\partial \mathcal{L}}{\partial \boldsymbol{z}_i}\frac{\partial \boldsymbol{z}_i}{\partial \boldsymbol{\Omega}_i} = \boldsymbol{\delta}_i\boldsymbol{n}_i$$

Once those effects of the weights and bias to the total loss have been calculated, we can update the weights doing the following transformations

$$\boldsymbol{\Omega}_i^{\text{new}} = \boldsymbol{\Omega}_i - \frac{\partial \mathcal{L}}{\partial \boldsymbol{\Omega}_i} \qquad \boldsymbol{b}_i^{\text{new}} = \boldsymbol{b}_i - \frac{\partial \mathcal{L}}{\partial \boldsymbol{b}_i}$$

This way, in the next iteration the performance of the network increases, as we have updated the weights to oppose their gradient. We could get to a point where there is no further improvement, but rather a stabilization or even in some situations a worsening of it. In this situation the model should stop learning as it has achieved its best performance.

There are many ways to initialize the weights at the beginning of training, for example using constant initialization, where all weights and biases are set to a constant (this could provoke all neurons in the network to learn the same features, which could lead to a symmetrical network, which in most cases is not a desired result, however it is common practice to set all biases to 0 initially). We can find a myriad of other initializations such as random states, where the values set to the weights are set according to a probability distribution, perhaps being the most common the Xavier (or Glorot) initialization, where the weights are initialized depending on the size of the layers, so that they from a uniform distribution. This way the variance during the forward pass and the gradient variance during the backward pass are preserved[4]. It is also worth noting some other initialization techniques as for example the orthogonal technique, where making use of orthogonal matrices and their properties, one can decouple the training time of the model from the depth of the network.
In reality the changes are not done by directly subtracting the gradient of the loss function, as making big changes to the weights on just one iteration could compromise the performance. Because of that, there is a hyperparameter (specified by the user, not learned) that controls how big of a change we allow the model to take in each iteration. It is known as the **learning rate** (η) and with it the changes are ruled by

$$\mathbf{\Omega}_i^{\text{new}} = \mathbf{\Omega}_i - \eta \frac{\partial \mathcal{L}}{\partial \mathbf{\Omega}_i} \qquad\qquad \boldsymbol{b}_i^{\text{new}} = \boldsymbol{b}_i - \eta \frac{\partial \mathcal{L}}{\partial \boldsymbol{b}_i}$$

The value of this hyperparameter influences the training, as a big value of η might drive big changes that could result on network errors, while a small value of η would cause tiny changes and increment the training time and cost. Typical values for η range between 0.1 and 0.0001.
The training technique described in here is known as **stochastic gradient descent (SGD)**.
This process is done for each of the datapoints on the dataset, and once all datapoints have been seen by the network, we say that the network has trained for one **epoch**. For the network to be specialized in the task we are training it for, it has to see all datapoints many times, often having to train between a few to thousands of epochs.
It is also common practice that instead of learning from one example at a time, the network looks at a set of examples before making any changes to its weights, this has some positive implications, such as the network training faster, learning better patterns and making steadier improvements. The set of datapoints that is passed together is known as a **batch**, and its cardinality is known as **batch size**. Batch sizes depend significantly on the task at hand, but typically range from 1 to

100.
To know whether the network has already trained or is still improving, often the loss value is monitored, and once it stops decreasing for a certain number of epochs, the network is considered to have trained, weights are stored as a checkpoint, and the model is ready to perform its task.
If the model is very expensive to train, it is not uncommon to download an already pretrained version of it and then **finetune** its parameters. Finetuning consists on making small modifications to the parameters of the model to better represent our data or get a better performance for our task. This way we can benefit from very large models without necessarily having the resources or time to be able to train them.
MLPs back themselves in the universal approximation theorem, which proves that convergence of the networks to a fitting function can be achieved. Nowadays there is a myriad of possibilities when picking the architecture of the neural network, many being much more sophisticated in terms of the mathematics involved. Some examples are graph neural networks (GNN), convolutional neural networks (CNN), recurrent neural networks (RNN), generative adversarial neural networks (GAN), etc.
Also there are some architectures that differ from that of the MLP, but can be used for the same purpose as MLPs, while others are fundamentally different and thought for other tasks.
Because to its close relation with mathematics and its possible application to sciences, an architecture worth considering are Kolmogorov-Arnold Networks (KAN). On the following subsection we aim to provide a detailed explanation of this architecture as well as a side-by-side comparison between MLPs and KANs.

3.1.5 KANs

Considering the Kolmogorov-Arnold representation theorem as the backbone of a neural network architecture has been on the minds of scientists for quite some time, but none of the previous attempts were able to generalize the theorem to achieve more complicated representations. This paradigm was changed in April 2024 when Liu et al. published a paper[19] presenting a novel architecture that made use of that theorem. The architecture was given the name of Kolmogorov-Arnold networks, to further emphasize the close relation to the theorem, and among the detailed description, analysis and convergence studies found in the paper, the authors theorized as those networks being a suitable substitute of MLPs. However, KANs present many fundamental differences with respect to MLPs, for example, instead of having predefined fixed non-linear activation functions on their nodes, and learnable weights on the edges, KANs learn their one-dimensional

"activation functions" on the edges of the network and on the nodes only summation is performed. The learnable function is typically taken a spline, even though some other approaches have been already popularized for example the use of Fourier series[20].

Architecture

According to Kolmogorov-Arnold representation theorem

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

it is enough to find functions Φ_q and $\varphi_{q,p}$ with which to represent the function. Provided there is no standardized way to get those functions, the network attempts to approximate them using much simpler ones, originally splines (as linear combination of B-splines), simplifying the problem to an optimization of the parameters of such combination. By doing this, the theorem loses generality, provoking that some functions can not be approximated with the same degree of success, for example it is the case for fractal functions or highly non-smooth ones.
The design of the KAN comes from interpreting the theorem as the complete expression of a neural network with three layers, with activation functions on the edges and summation on the nodes. It corresponds to an input layer (width n), an intermediate layer (width $2n + 1$) and output layer (width 1). As the activation functions are imposed to be a linear combination of B-splines, there is a loss of generality in the representation of functions. This can be compensated by making the layer wider and deeper (such as in the case of MLPs). To develop the generalization more easily we can make use of matrix formalism to represent the KAN.
A KAN layer i has a number of neurons N_i and on each of those neurons only summation of the preactivated values is performed; after the summation, each neuron feeds to the following N_{i+1} neurons, through edges that contain the activation functions, the process of feeding forward can be interpreted as a matrix of one-dimensional functions $\boldsymbol{\Phi}_i = \{\phi^i_{p,q}\}$ with $p = 1, \cdots, N_i$ and $q = 1, \ldots, N_{i+1}$. Each of the functions $\phi^i_{p,q}$ has trainable parameters.
With this formalism it is easy now to see how the theorem is the expression of a three layer KAN, in detail

$$f(\boldsymbol{x}) = \underbrace{\sum_{q=1}^{2n+1} \Phi_q \left(\underbrace{\sum_{p=1}^{n} \varphi_{q,p}(\underbrace{x_p}_{\text{Input layer}})}_{\text{Hidden layer 1}} \right)}_{\text{Output layer}}$$

We can express the shape of a KAN by means of an ordered set of integers $(N_1 \dots, N_L)$ where N_i is the width of the ith layer. As a MLP, a KAN is a feed-forward fully connected neural network, hence meaning that in between layer i and layer $i+1$ there are $N_i N_{i+1}$ edges (which in the case of KANs correspond to activation functions). We denote the function that connects the j-th neuron of the i layer to the k-th neuron of the $i+1$ layer as $\phi^i_{j,k}$. On an analogous manner as in the MLP section, we denote n^i_j the value that the neuron j in layer i, and $\boldsymbol{n}_i$ all the neurons in layer i. With this convention we can express the k-th neuron on the following layer as

$$n^{i+1}_k(\boldsymbol{n}_i) = \sum_{j=1}^{N_i} \phi^i_{j,k}(n^i_j),\ j = 1, \dots, N_{i+1}$$

Alternatively we can express this forward step for all neurons at once, using matrix notation as

$$\boldsymbol{n}_{i+1} = \begin{pmatrix} n^{i+1}_1 \\ \vdots \\ n^{i+1}_{N_{i+1}} \end{pmatrix} = \begin{pmatrix} \phi^i_{1,1} & \cdots & \phi^i_{N_i,1} \\ \vdots & & \vdots \\ \phi^i_{1,N_{i+1}} & \cdots & \phi^i_{N_i,N_{i+1}} \end{pmatrix} \begin{pmatrix} n^i_1 \\ \vdots \\ n^i_{N_i} \end{pmatrix} = \boldsymbol{\Phi}_i \boldsymbol{n}_i$$

As in the case of MLPs we can deepen the network to achieve better performance, and in that situation the KAN expression is

$$\hat{\boldsymbol{y}} = \mathrm{KAN}(\boldsymbol{x}) = \boldsymbol{n}_L \circ \boldsymbol{n}_{L-1} \circ \cdots \circ \boldsymbol{n}_1 = (\boldsymbol{\Phi}_{L-1} \circ \boldsymbol{\Phi}_{L-2} \circ \cdots \circ \boldsymbol{\Phi}_1)\boldsymbol{n}_1$$

and dropping the matrix form to get an expression analogous to that of the theorem, we get

$$\hat{\boldsymbol{y}}_k = \mathrm{KAN}(\boldsymbol{x})_k = \sum_{j_{L-1}=1}^{N_{L-1}} \phi^{L-1}_{j_{L-1},k} \left(\sum_{j_{L-2}=1}^{N_{L-2}} \phi^{L-2}_{j_{L-2},j_{L-1}} \left(\cdots \left(\sum_{j_2=1}^{N_2} \phi^2_{j_2,j_3} \left(\sum_{j_1=1}^{N_1} \phi^1_{j_1,j_2}(x_{j_1}) \right) \right) \cdots \right) \right)$$

And once more, we can observe from here that the theorem is a $(n, 2n+1, 1)$-KAN. Also, in here it is emphasized how the output of this architecture can be multidimensional

We can recall that the MLP architecture can be expressed as

$$\mathrm{MLP}(\boldsymbol{x}) = (\sigma_{L-1} \circ \boldsymbol{W}_{L-1} \circ \sigma_{L-1} \circ \boldsymbol{W}_{L-2} \circ \cdots \circ \sigma_2 \circ \boldsymbol{W}_2 \circ \sigma_1 \circ \boldsymbol{W}_1)\boldsymbol{x}$$

Implementation of KANs

Once the architecture of a KAN has been described, we can concern ourselves with their training, properties and limitations. We can observe that since all operations involved in the construction of KANs are differentiable we can make use of both

SGD and backpropagation algorithms to train those models (equally as we did with MLPs).
So far we have avoided detailing the explicit structure of the activation functions in KANs, as well as their trainable parameters. The original authors propose $\phi^i_{j,k}$ to be a combination of a basis function and spline function, mathematically

$$\phi^i_{j,k} = \alpha^i_{j,k} s^i_{j,k}(x) + \beta^i_{j,k} b^i_{j,k}(x)$$

Where $b(x)$ denotes a *basis function*, normally a sigmoid linear unit (SiLU)

$$b^i_{j,k}(x) = \text{SiLU}(x) = \frac{x}{1 + e^{-x}}$$

and $s(x)$ is a linear combination of b-splines

$$s^i_{j,k}(x) = \sum_{l=0}^{K-(d+1)} c^i_{j,k,l} B_{l,d}(x)$$

with $K + 1$ being the length of the extended knot vector and d being the degree of the used b-splines. Of those functions, the coefficients $c^i_{j,k,l}$, $\alpha^i_{j,k}$ and $\beta^i_{j,k}$ are trainable (note that $\alpha^i_{j,k}$ and $\beta^i_{j,k}$ are redundant, as they could be absorbed by the other coefficients, but they are kept for clarity).
In here if another function other than b-splines was to be used, $s(x)$ should be replaced with it. For example this is what is done in the aforementioned Fourier KANs.
The initialization of the coefficients of b-splines is taken according to a normal distribution centred at 0 and with a small variance, in the case of the paper it is taken $\mathcal{N}(\mu, \sigma^2) = \mathcal{N}(0, 0.01)$. This way the spline is approximately 0, but we avoid the problems mentioned in the previous subsection. Also the weights are initialized to be $\alpha^i_{j,k} = 1$ and $\beta^i_{j,k}$ is taken from a uniform distribution with mean 0 and standard deviation $\sigma = \sqrt{2/(N_i + N_{i+1})}$ (Xavier initialization).
Another thing to consider is the fact that the spline has support only on a definite region set by its knot vector, but sometimes the output of the previous layer can fall outside the support of the spline. In order to avoid this, the knots of the spline can also be learnt (preferred approach) or after each layer the output of the neuron is normalized to fall within the range (this yields lower performance, so it's not common practice).

Comparison with MLPs

The training time and computational cost of the model is directly given by the amount of trainable parameters that a given model has. In order to ease the

parameter count of a KAN we make the following considerations, we assume a network with depth L constant width N, and all splines of degree d (typically $d = 3$), and with a knot vector consisting of $K + 1$ distinct knots.
As we know that there are N^2 ϕ-functions per layer, the amount of functions in the total network is N^2L, and each of the functions has $K + d$, by considering all intermediate knots to be only repeated once, that is with the maximum smoothness possible for all used splines and repeating the extremal knots d times we have a total number of parameters of $K + d - 1$ (by the amount of distinct B-splines satisfying this conditions), and accounting for the weights β (we assume α to be trained by the model we end up having a total number of parameters)

$$\mathcal{O}(N^2L(K + d)) \sim \mathcal{O}(N^2LK)$$

as typically $d \ll K$. MLPs on the other hand require $\mathcal{O}(N^2L)$ parameters. A priori it seems that MLPs are more efficient than KANs, but due to the construction of KANs, those require significantly smaller values of N, which ends up improving the performance and training time, also it generalizes better and facilitates interpretability.
Another advantage that KANs exhibit over MLPs is that their performance can be enhanced mid-training by extending the knot vector of intermediate splines. Splines approximate better any function if their knot vector is larger, (we can think of this as in the limit having one knot for each point of the function). MLPs don't have any process similar as this one, and the way to improve their performance is either widen or deepen the network, two situations that are often much more expensive than incrementing the knot vector length.
We can understand this process as follows, given a one-dimensional function in a bounded region $[a, b]$, which is approximated by two different splines of degree d with respective knot vectors $\boldsymbol{t} = \{a = t_0, \ldots, t_{K'} = b\}$ and $\boldsymbol{t}' = \{a = t'_0, \ldots, t'_{K'}\}$ with $K' > K$, we have the representations

$$f_{\text{coarse}}(x) = \sum_{i=0}^{K-(d+1)} c_i B_{i,d}(x) \qquad f_{\text{fine}}(x) = \sum_{i=0}^{K'-(d+1)} c'_i B_{i,d}(x)$$

Knowing the coefficients $\{c_i\}$ for the coarse representation we can get the coefficients for the finer representation by considering a distribution of x and minimizing the expected value of the difference between the two, this is

$$\{c'_j\} = \underset{\{c'_j\}}{\operatorname{argmin}} \left[\mathbb{E}\left(f_{\text{fine}} - f_{\text{coarse}}\right)\right] = \underset{\{c'_j\}}{\operatorname{argmin}} \left[\mathbb{E}\left(\sum_{i=0}^{K'-(d+1)} c'_i B_{i,d}(x) - \sum_{i=0}^{K-(d+1)} c_i B_{i,d}(x)\right)\right]$$

This can be done via minimization algorithms for example the least squares one. All knot vectors of splines can be extended independently.

The effect that this process has on training can be seen when monitoring loss curves, typically the profile of a loss function over time (epoch iteration) is a smooth asymptotically decreasing curve (something similar to the positive branch of $1/x$), with the knot extension technique the profile is more similar to a staircase shape, where whenever the rate of improvement slows down, we can trigger a faster convergence by improving the knot vector.

Sparsity and pruning

Sometimes finding the optimal network structure (depth, width of each layer, etc.) can be challenging, and to overcome this issue it is typical to train the network with sparsity, and then prune the neurons that are deemed unimportant.
On MLPs sparsely training can be achieved by using a L_1 regularization of linear weights (Known as Lasso). That is not enough for KANs, and the training has to be done with a L_1-norm term and an entropy related term. Mathematically we can define the L_1 norm of an activation function ϕ is the average magnitude of the function over all the S samples that it sees.

$$\|\phi\|_1 = \frac{1}{S}\sum_{s=1}^{S} |\phi(x^{(s)})|$$

where $x^{(s)}$ is the s-th sample. If we consider a whole layer Φ_i in the network, we can define its L_1 norm as

$$\|\Phi_i\|_1 = \sum_{j=1}^{N_i}\sum_{k=1}^{N_{i+1}} \|\phi_{j,k}^i\|_1$$

and also we can define the entropy term (analogous to Shannon entropy) as

$$S(\Phi) = -\sum_{j=1}^{N_i}\sum_{k=1}^{N_{i+1}} \frac{\|\phi_{j,k}^i\|_1}{\|\Phi_i\|_1} \log\left(\frac{\|\phi_{j,k}^i\|_1}{\|\Phi_i\|_1}\right)$$

And now in order to train KANs sparsely we define the loss function

$$\mathcal{L} = \mathcal{L}_{\text{prediction}} + \lambda\left(\mu_1 \sum_{i=1}^{L-1} \|\Phi_i\|_1 + \mu_2 \sum_{i=1}^{L-1} S(\Phi_i)\right)$$

where $\mathcal{L}_{\text{prediction}}$ is the loss for the task we are training, μ_1, μ_2 are relative weights, usually set to 1 and λ controls the strength of the regularization.
In order to determine whether the j-th neuron of the i layer should be preserved or pruned, we define its input and output as follows

$$I_j^i = \max_{k\in[1..N_{i-1}]} \|\phi_{k,j}^{i-1}\|_1 \qquad\qquad O_j^i = \max_{k\in[1..N_i]} \|\phi_{j,k}^i\|_1$$

If both I_j^i and O_j^i are below a typical threshold θ (typically $\theta = 10^{-2}$), the neuron is considered unimportant and is then pruned from the network.

Following this methodology, we end up with the smallest possible network that correctly explains the behaviours of the points. This apparently innocent statement about KANs gives us insight about the mathematics underlying the problem we are tackling. Additionally KANs can yield a mathematical expression of the function they have converged to.

There are many more advantages that KANs present with respect to MLPs, for example they don't exhibit catastrophic forgetting, have a theoretically faster rate of convergence, can achieve better approximations or have good local properties that bound the changes in the event of a weight modification, but those won't be covered in this work, check [19] for further information about the architecture.

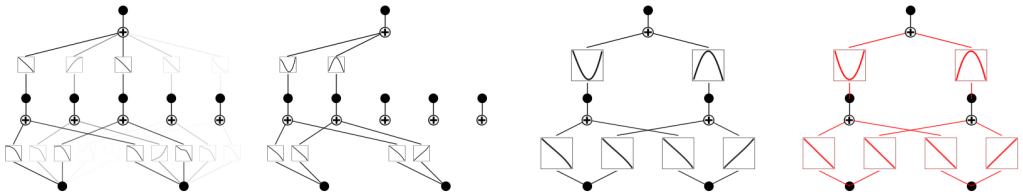
Other uses to KANs in science

In this small final subsection we want to illustrate some uses and examples of KANs in various domains of science.

One first example could be how to represent the function $f(x_1, x_2) = x_1 x_2$ using only summation and functions of a single variable. We want the KAN to discover the way to do this operation, so we start with a general $(2, 5, 1)$ KAN and initialize it randomly. For the training we generate a train-test dataset with 1000 datapoints each for values of x_1, x_2 ranging between -1 and 1, the points of the dataset are random. Later the model is trained with sparsity and neurons that are deemed unimportant get pruned from the network. After the model has converged to a structure, we train it further and refine the knot vector (from 3 intermediate knots to 5) to achieve a finer result. Once we have reached a point where the loss stops improving, we query the model to check which symbolic expression is closer to the approximated spline.

On our case the library of symbolic functions considered all monomials with exponents ranging from -5 to 5 (in jumps of 1), monomials with exponent $1/2$ (square root) and $3/2$ (square root cubed), exponential and logarithm functions and trigonometric and hyperbolic functions. When the symbolic functions are determined, a search for the optimal affine parameters is conducted by the model being able to achieve machine precision for the description of the data. That is, considering the symbolic function $f(x)$, the model tries to find $a, b, c, d \in \mathbb{R}$ so that the function $af(bx + c) + d$ better describes the data.

The value to which it converges to is displayed and can be compared experimentally. For this case, all symbolic fits were on point, and the final analytical



(a) (b) (c) (d)

Figure 3.1: (a) Random initialization of the $(2,5,1)$ KAN (b) Model state after training for 50 steps, we can see that most neurons have succumbed to the penalty and aren't used, exposing a $(2,2,1)$ structure (c) Model after further 100 steps of training and refinement of grid from 3 to 5 knots (d) Final model when reached machine precision with correct symbolic functions

.

expression that was returned was

$$0.26(0.99x_1 + 0.95x_2)^2 - 0.21(1.11x_1 - 1.06x_2)$$

Even though the coefficients are a bit far from one, we can see what the model is attempting to do, it wants to get the product x_1x_2 from the identities of square of a sum/ difference. Indeed,

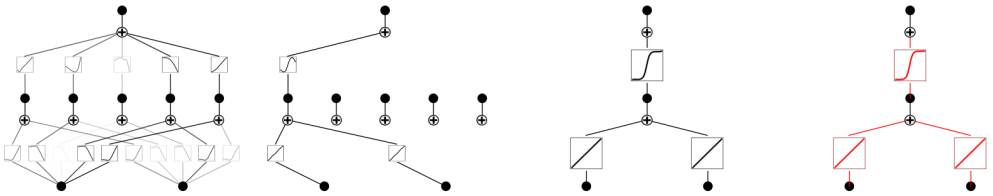
$$\frac{1}{4}(x_1 + x_2)^2 - \frac{1}{4}(x_1 - x_2)^2 = \frac{1}{4}\left(x_1^2 + 2x_1x_2 + x_2^2 - (x_1^2 - 2x_1x_2 + x_2^2)\right) = x_1x_2$$

We attributed this error in the coefficients to a small training dataset, so we repeated the after-symbolism training with a larger dataset of $(\text{train}, \text{test}) = (1e5, 5e4)$ datapoints ranging from -100 to 100. With those datapoints the model converged to a formula whose expansion was virtually x_1x_2 (as any other term was smaller than 10^{-13}).

The whole process of this model training run can be seen in figure 3.1.5, in particular from 3.1.5(d) we can observe the graphical expression of the formula (without coefficients and considering a global minus sign inside the parentheses) when moving from bottom to top and calling x_1 to the left node and x_2 to the right one. Another example could be trying to see if an expression can be thought of in a more efficient way, for example, let us consider the function

$$f(x_1, x_2) = \frac{exp(x_1 + x_2) - 1}{\exp(x_1 + x_2) + 1}$$

Again we repeat the same procedure as in the previous case, which results in the situations in figure 3.1.5. In this case it was not necessary to increase the size of the dataset.



(a) (b) (c) (d)

Figure 3.2: (a) Random initialization of the $(2,5,1)$ KAN (b) Model state after training for 50 steps, we can see that most neurons have succumbed to the penalty and aren't used, revealing a $(2,1,1)$ structure (c) Model after further 50 steps of training and refinement of grid from 3 to 5 knots, a tanh function can be clearly observed now in the top edge (d) Final model when reached machine precision with correct symbolic functions

.

When achieving machine precision with the symbolic the expression that the model returned was

$$1.0tanh(0.5x_1 + 0.5x_2)$$

allowing us to deal with the original function by means of the hyperbolic tangent rather than using a quotient of exponentials.
We can validate this expression by observing that

$$tanh(x_1 + x_2) = \frac{sinh(x_1 + x_2)}{cosh(x_1 + x_2)} = \frac{\frac{exp(x_1+x_2)-exp(-x_1-x_2)}{2}}{\frac{exp(x_1+x_2)+exp(-x_1-x_2)}{2}} = \frac{exp(x_1 + x_2) - 1}{exp(x_1 + x_2) + 1}$$

Where in the last equality we have multiplied by $exp(x + y)$ both on the numerator and on the denominator.
This type of example manifests that KANs are able to determine the minimal mathematical expressions required to deal with a given problem.

Conclusion

Departing from the formal academic tone present in the whole project, I can state that I have learnt much more than what I originally expected. What I first thought that was to be a mere procedure required for the completion of the bachelors degree, turned out to be much more interesting at the end. The effort of studying an AI architecture to its most fundamental theorems morfs into a sense of reward. After comprehending what is going on at the groundest of levels, one can only anticipate with illusion and joy all architectures that are to come.

Throughout this project, we have attempted to cover approximately all the mathematics involved in the confection of a new AI architecture. We have observed how a classical theorem proven more than seventy years ago can re-emerge to rebuild (and improve) one of the building blocks of artificial intelligence. This is of value not just because it shows how a real analysis theorem is applied to another domain of science, but also as it allows one to have a glimpse of future research. Additionally it allows us to dream about how existing mathematics (or mathematics to come) will shape the tomorrow world.

There is no doubt that the world is changing rapidly, and the insertion of artificial intelligence in our society comes with large implications. This accelerating rate of change is only possible with a parallel improvement in mathematical methods and results. Once more the research in this realm of science sets the stepping stone on what is to be built in incoming years.

The challenges that come with this change are not only of a mathematical fashion, as all other disciplines are affected by it. It also poses great challenges in developmental economics, behavioural dynamics, psychology of the mind, philosophy of our identity, etc. Only through pertinent research we will be able to overcome all difficulties and be able to proudly look back into this uncertain time.

As a final note I would like to add that different circumstances have made me align this project with the world of AI, but mathematics would have proven equally gratifying when used in any other field, that alone suffices the completion of the degree. In summary, through mathematics any black boxes become less opaque, and any field can become transparent to one's sight.

Appendix A

Real analysis theorems

A.1 Heine-Cantor theorem

Theorem A.1.1 (Heine-Cantor). *If (X, d_X) and (Y, d_Y) are metric spaces, $f : X, Y$ is a continuous function and X is compact, then f is uniformly continuous*

We require some definitions prior of proving the theorem

Definition A.1.1. Let X be a metric space with distance d_X, given a point $x \in X$ and a number $r \in \mathbb{R}$ we denote by $B(x, r)$ the **open ball with centre x and radius r**, i.e. the open set $\{y \in X : d_X(x, y) < r\} \subseteq X$

Definition A.1.2. Let X be a metric space with distance d_X, given a point $x \in X$ and a number $r \in \mathbb{R}$ we denote by $\bar{B}(x, r)$ the **closed ball with centre x and radius r**, i.e. the closed set $\{y \in X : d_X(x, y) \leq r\} \subseteq X$

Definition A.1.3. Let X and Y be metric spaces with respective distances d_X and d_Y, then a function $f : X \to Y$ is uniformly continuous if $\forall \varepsilon > 0$ exists a $\delta > 0$ (with only dependence on ε) such that $\forall x, y \in X$ that satisfy $d_X(x, y) < \delta$ it is satisfied that $d_Y(f(x), f(y)) < \varepsilon$

Definition A.1.4. A topological space X is **compact** if every open cover of X has a finite subcover

Proof of Heine-Cantor theorem. Given a value of $\varepsilon > 0$, by continuity of f we know that for any $x, y \in X$, $\exists \delta_x$ such that if $d_X(x, y) < \delta_x$, then $d_Y(f(x), f(y)) < \varepsilon/2$.
We can now consider the union

$$\bigcup_{x \in X} B\left(x, \frac{\delta_x}{2}\right)$$

As it is considering all points of X, and it is a union of open sets, we have that it is an open cover of X. As X is compact, we can find a finite subcover

$$\bigcup_{i=1}^{m} B\left(x_i, \frac{\delta_{x_i}}{2}\right)$$

We can now take $\delta = \min_{i \in [1..m]} \delta_{x_i} > 0$. Then for any $x, y \in X$ such that $d_X(x,y) < \delta$ we can find an element from the subcover such that contains x. We denote it $B\left(x_i, \delta_{x_i}/2\right)$ so we have that

$$d_X(x_i, y) \leq d_X(x_i, x) + d_X(x, y) < \frac{\delta_{x_i}}{2} + \delta \leq \delta_{x_i}$$

So by definition we have that $d_Y(f(x_i), f(y)) < \varepsilon/2$ and $d_Y(f(x), f(y)) < \varepsilon/2$, therefore by triangular inequality we have that

$$d_Y(f(x), f(y)) \leq d_Y(f(x), f(x_i)) + d(f(x_i), f(y)) < \frac{\varepsilon}{2} + \frac{\varepsilon}{2} = \varepsilon$$

□

A.2 Weierstrass approximation theorems

In order to prove the Weierstrass rational approximation theorem, we first prove the Weierstrass approximation theorem (over the real field), and then prove the rational version almost as a corollary

Theorem A.2.1 (Weierstrass rational approximation)**.** *For any continuous function $f : [0,1] \to \mathbb{R}$ and any $\varepsilon > 0$, exists a polynomial $g \in \mathbb{Q}[x]$ (a polynomial of rational coefficients) such that $\forall x \in [0,1]$ we have*

$$|f(x) - g(x)| < \varepsilon$$

Theorem A.2.2 (Weierstrass approximation)**.** *For any continuous function $f : [0,1] \to \mathbb{R}$ and any $\varepsilon > 0$, exists a polynomial g such that $\forall x \in [0,1]$ we have*

$$|f(x) - g(x)| < \varepsilon$$

In order to prove Weierstrass theorem, we can follow Bernstein's constructive proof, for which we need some definitions

Definition A.2.1. A **Bernstein polynomial** is defined as

$$Be_{n,k}(x) = \binom{n}{k} x^k (1-x)^{n-k}, \text{ for } k \in [0..n]$$

where

$$\binom{n}{k} = \frac{n!}{k!(n-k)!}$$

Observation. $Be_{n,k}(x) \geq 0$ for all n, for all $k \in [0..n]$, for all $x \in [0,1]$

Lemma A.2.1.

$$\sum_{k=0}^{n} \left[\prod_{j=0}^{p-1}(k-j)\right] Be_{n,k}(x) = \left[\prod_{j=0}^{p-1}(n-j)\right] x^p$$

Proof.

$$\sum_{k=0}^{n} \left[\prod_{j=0}^{p-1}(k-j)\right] Be_{n,k}(x) = \sum_{k=0}^{n} \left[\prod_{j=0}^{p-1}(k-j)\right] \frac{n!}{k!(n-k)!} x^k (1-x)^{n-k}$$

Now we can observe that the first p terms of the sum are going to be exactly zero, so we can directly omit those indices

$$\sum_{k=0}^{n} \left[\prod_{j=0}^{p-1}(k-j)\right] \frac{n!}{k!(n-k)!} x^k (1-x)^{n-k} = \sum_{k=p}^{n} \left[\prod_{j=0}^{p-1}(k-j)\right] \frac{n!}{k!(n-k)!} x^k (1-x)^{n-k}$$

$$= \sum_{k=p}^{n} \frac{n!}{(k-p)!(n-k)!} x^k (1-x)^{n-k}$$

And now by making a translation ($k = l + p$) on the index we have

$$\sum_{k=p}^{n} \frac{n!}{(k-p)!(n-k)!} x^k (1-x)^{n-k} = \sum_{l=0}^{n-p} \frac{n!}{((l+p)-p)!(n-(l+p))!} x^{l+p} (1-x)^{n-(l+p)}$$

$$= \sum_{l=0}^{n-p} x^p \frac{n!}{(n-p)!} \frac{(n-p)!}{l!((n-l)+p)!} x^l (1-x)^{(n-l)+p}$$

And now we can observe that the second term of the product is the binomial expansion of $(x + (1-x))^{n-p}$, hence

$$x^p \frac{n!}{(n-p)!} \sum_{l=0}^{n-p} \frac{(n-p)!}{l!((n-l)+p)!} x^l (1-x)^{(n-l)+p} = x^p \frac{n!}{(n-p)!} \left(x + (1-x)\right)^{n-p}$$

$$= x^p \frac{n!}{(n-p)!} = \left[\prod_{j=0}^{p-1}(n-j)\right] x^p$$

□

Lemma A.2.2.

$$\sum_{k=0}^{n} \left(\frac{k}{n} - x\right)^2 Be_{n,k}(x) = \frac{x(1-x)}{n}$$

Proof. We can see this result by expanding the square in each term

$$\sum_{k=0}^{n}\left(\frac{k}{n}-x\right)^2 Be_{n,k}(x)=\sum_{k=0}^{n}\left(\frac{k^2}{n^2}-\frac{2kx}{n}+x^2\right)Be_{n,k}(x)$$

$$=\frac{1}{n^2}\sum_{k=0}^{n}k^2Be_{n,k}(x)-\frac{2x}{n}\sum_{k=0}^{n}kBe_{n,k}(x)+x^2\sum_{k=0}^{n}Be_{n,k}(x)$$

Now we can express k^2 as the difference

$$k^2=k(k-1)+k$$

And substituting we obtain

$$\sum_{k=0}^{n}\left(\frac{k}{n}-x\right)^2 Be_{n,k}(x)=\frac{1}{n^2}\sum_{k=0}^{n}k(k-1)Be_{n,k}(x)+\left(-\frac{2x}{n}+\frac{1}{n^2}\right)\sum_{k=0}^{n}kBe_{n,k}(x)+x^2\sum_{k=0}^{n}Be_{n,k}(x)$$

Now we can apply the lemma A.2.1, and we get

$$\sum_{k=0}^{n}\left(\frac{k}{n}-x\right)^2 Be_{n,k}(x)=\frac{1}{n^2}n(n-1)x^2+\left(-\frac{2x}{n}+\frac{1}{n^2}\right)nx+x^2=\frac{(n-1)x^2}{n}+\left(-2x^2+\frac{x}{n}\right)+x^2$$

$$=x^2-\frac{x^2}{n}-2x^2+\frac{x}{n}+x^2=\frac{x(1-x)}{n}$$

□

Observation. An alternative proof of lemma A.2.2 could be developed by considering the variance of a binomial distribution. For that we should consider $Be_{n,k}$ to be the probabilities of the events $\frac{k}{n}$.

Proof of Weierstrass theorem. We can consider the polynomial

$$Be_n^f(x)=\sum_{k=0}^{n}f\left(\frac{k}{n}\right)Be_{n,k}(x) \tag{A.1}$$

This polynomial can be interpreted as a weighted Riemann sum, we want to see that given $\varepsilon>0$, we can find a positive integer N such that $\forall n>N$ and $\forall x\in[0,1]$, we have

$$|B_n^f(x)-f(x)|<\varepsilon$$

As f is continuous on the closed interval, it is uniformly continuous (Theorem A.1.1), so $\forall\varepsilon>0$ we can find a positive real number $\delta=\delta(\varepsilon)$ satisfying that if $|x-y|<\delta$, then

$$|f(x)-f(y)|<\frac{\varepsilon}{2}$$

According to lemma A.2.1, we know that

$$\sum_{k=0}^{n} B_{n,k}(x) = 1$$

So we can do the following

$$Be_n^f(x) - f(x) = \sum_{k=0}^{n} \left[f\left(\frac{k}{n}\right) Be_{n,k}(x) \right] - f(x) \sum_{k=0}^{n} Be_{n,k}(x)$$

$$= \sum_{k=0}^{n} f\left(\frac{k}{n}\right) Be_{n,k}(x) - \sum_{k=0}^{n} f(x) Be_{n,k}(x) = \sum_{k=0}^{n} \left[f\left(\frac{k}{n}\right) - f(x) \right] Be_{n,k}(x)$$

And now we can separate the sum according to the triangular inequality as follows

$$|Be_n^f(x) - f(x)| \leq \sum_{k:\left|\frac{k}{n}-x\right|<\delta} |Be_n^f(x) - f(x)| + \sum_{k:\left|\frac{k}{n}-x\right|\geq\delta} |Be_n^f(x) - f(x)|$$

Now rewriting $B_n^f(x) - f(x)$ as above, and taking into account that all Bernstein polynomials are non-negative we have

$$|Be_n^f(x) - f(x)| \leq \sum_{k:\left|\frac{k}{n}-x\right|<\delta} \left| f\left(\frac{k}{n}\right) - f(x) \right| Be_{n,k}(x) + \sum_{k:\left|\frac{k}{n}-x\right|\geq\delta} \left| f\left(\frac{k}{n}\right) - f(x) \right| Be_{n,k}(x)$$

and because of the uniform continuity of f

$$|Be_n^f(x) - f(x)| \leq \frac{\varepsilon}{2} \sum_{k:\left|\frac{k}{n}-x\right|<\delta} Be_{n,k}(x) + 2||f|| \sum_{k:\left|\frac{k}{n}-x\right|\geq\delta} Be_{n,k}(x)$$

And now we can study the second term

$$\sum_{k:\left|\frac{k}{n}-x\right|\geq\delta} Be_{n,k}(x) = \sum_{k:\left|\frac{k}{n}-x\right|\geq\delta} \frac{\left|\frac{k}{n}-x\right|^2}{\left|\frac{k}{n}-x\right|^2} Be_{n,k}(x) \leq \frac{1}{\delta^2} \sum_{k:\left|\frac{k}{n}-x\right|\geq\delta} \left|\frac{k}{n}-x\right|^2 Be_{n,k}(x)$$

And now using lemma A.2.2 we have that

$$\sum_{k:\left|\frac{k}{n}-x\right|\geq\delta} Be_{n,k}(x) \leq \frac{1}{\delta^2} \frac{x(1-x)}{n}$$

Now considering

$$\sum_{k:\left|\frac{k}{n}-x\right|<\delta} Be_{n,k}(x) \leq 1$$

by lemma A.2.1, it is seen that

$$|Be_n^f(x) - f(x)| \leq \frac{\varepsilon}{2} + 2\|f\|\frac{1}{\delta^2}\frac{x(1-x)}{n} \leq \frac{\varepsilon}{2} + \frac{\|f\|}{2n\delta^2}$$

Where we have used that $\frac{x(1-x)}{n}$ has a maximum value of $\frac{1}{4}$. We can conclude the proof by solving the above inequality for n, and observing that considering any value

$$n > \|f\|\frac{1}{\varepsilon\delta^2}$$

the theorem is satisfied. □

Proof of Weierstrass rational theorem. Given a polynomial $p = p_n x^n + \cdots + p_1 x + p_0$, we can measure its distance to another polynomial $q = q_n x^n + \cdots + q_1 x + q_0$ as

$$\begin{aligned}
\|p - q\| &= \sup_{x\in[0,1]} |p(x) - q(x)| \leq \sup_{x\in[0,1]} |p_n x^n + \cdots + p_0 - (q_n x^n + \cdots + q_0)| \\
&= \sup_{x\in[0,1]} |(p_n - q_n)x^n + \cdots + (p_0 - q_0)| \leq \sup_{x\in[0,1]} (|p_n - q_n|x^n + \cdots + |p_0 - q_0|) \\
&= \sup_{x\in[0,1]} \sum_{k=0}^{n} |p_k - q_k|x^k \leq \sum_{k=0}^{n} |p_k - q_k|
\end{aligned}$$

Now, since rational numbers are dense on the real line, we can consider the coefficient q_i to be at distance $\varepsilon/2n$ from p_i, this way, we have that $\|p - q\| \leq \varepsilon/2$. We can now prove the theorem using triangular inequality

$$\|q - f\| \leq \|q - p\| + \|p - f\| < \frac{\varepsilon}{2} + \frac{\varepsilon}{2} = \varepsilon$$

□

Bibliography

[1] A. Kolmogorov. "On the representation of continuous functions of several variables as superpositions of continuous functions of a smaller number of variables". In: *Doklady Akademii Nauk SSSR (Proceedings of the USSR Academy of Sciences)* 114.5 (1957). Translated and reprinted, also available online, pp. 953–956. URL: https://cs.uwaterloo.ca/~y328yu/classics/Kolmogorov57.pdf.

[2] Carl de Boor. *A Practical Guide to Splines*. Revised edition. Madison, WI: Springer-Verlag, 2001.

[3] Cambridge Numerical analysis. *B-splines*. Approximation theory - Lectures 12, 13, 14. Lent term 2005. URL: https://www.damtp.cam.ac.uk/user/na/PartIIIat/b12.pdf, https://www.damtp.cam.ac.uk/user/na/PartIIIat/b13.pdf, https://www.damtp.cam.ac.uk/user/na/PartIIIat/b14.pdf.

[4] X. Glorot and Y. Bengio. "Understanding the difficulty of training deep feedforward neural networks". In: *Proceedings of Machine Learning Research* 9 (2010), pp. 249–256. URL: https://proceedings.mlr.press/v9/glorot10a/glorot10a.pdf.

[5] Michael Loss. *The Weierstrass approximation theorem*. Course notes. Fall 2016. URL: https://loss.math.gatech.edu/16FALLTEA/NOTES/weierstrassbernstein.pdf.

[6] Pragati Baheti. *Train Test Validation Split: How To & Best Practices*. 2021. URL: https://www.v7labs.com/blog/train-validation-test-set (visited 24/10/2025).

[7] S. Dzhenzher and A. Skopenkov. *A structured proof of Kolmogorov's Superposition Theorem*. 2022. arXiv: 2105.00408 [math.FA]. URL: https://arxiv.org/abs/2105.00408.

[8] Michael S. Floater. *Marsden's identity and linear independence of B-splines*. MAT4170 Course notes. Mar. 2022. URL: https://www.uio.no/studier/emner/

matnat/math/MAT4170/v22/undervisningsmateriale/marsden_lin_indep.pdf.

[9] Samy Baladram. *Multilayer Perceptron, Explained: A Visual Guide with Mini 2D dataset*. 2024. URL: https://towardsdatascience.com/multilayer-perceptron-explained-a-visual-guide-with-mini-2d-dataset-0ae8100c5d1c/ (visited 24/10/2025).

[10] Wikipedia contributors. *Activaiton function*. 2025. URL: https://en.wikipedia.org/wiki/Activation_function (visited 24/10/2025).

[11] Wikipedia contributors. *B-spline*. 2025. URL: https://en.wikipedia.org/wiki/B-spline (visited 6/10/2025).

[12] Wikipedia contributors. *Baire category theorem*. 2025. URL: https://en.wikipedia.org/wiki/Baire_category_theorem (visited 9/10/2025).

[13] Wikipedia contributors. *Baire space*. 2025. URL: https://en.wikipedia.org/wiki/Baire_space (visited 9/10/2025).

[14] Wikipedia contributors. *Dense set*. 2025. URL: https://en.wikipedia.org/wiki/Dense_set (visited 9/10/2025).

[15] Wikipedia contributors. *Heine–Cantor theorem*. 2025. URL: https://en.wikipedia.org/wiki/Heine%E2%80%93Cantor_theorem (visited 9/10/2025).

[16] Wikipedia contributors. *Kolmogorov-Arnold representation theorem*. 2025. URL: https://en.wikipedia.org/wiki/Kolmogorov%E2%80%93Arnold_representation_theorem (visited 9/10/2025).

[17] Wikipedia contributors. *Spline (mathematics)*. 2025. URL: https://en.wikipedia.org/wiki/Spline_(mathematics) (visited 6/10/2025).

[18] Wikipedia contributors. *Universal approximation theorem*. 2025. URL: https://en.wikipedia.org/wiki/Universal_approximation_theorem (visited 24/10/2025).

[19] Ziming Liu et al. *KAN: Kolmogorov-Arnold Networks*. 2025. arXiv: 2404.19756 [cs.LG]. URL: https://arxiv.org/abs/2404.19756.

[20] Jusheng Zhang et al. *Kolmogorov-Arnold Fourier Networks*. 2025. arXiv: 2502.06018 [cs.LG]. URL: https://arxiv.org/abs/2502.06018.