Universal Approximation Theorem

Eslam Khalaf

July 4, 2026

1 Abstract

This paper serves as an introduction to universal approximation theorems (UATs). UATs demonstrate how powerful neural networks (NNs) are in approximating functions. They state that, in theory, any function could be approximated with any required degree of accuracy due to the structure of neural networks

2 Introduction

Neural networks have become one of the most powerful tools in modern machine learning, capable of solving problems ranging from simple logical operations to complex tasks like image recognition and natural language processing. But this raises a fundamental theoretical question: what exactly are neural networks capable of representing? Is there a fixed limit to the kinds of functions they can learn, or can they, given the right configuration, approximate virtually any function we care about?

This question is answered by a family of results known as the Universal Approximation Theorems (UATs). Informally, these theorems state that a feedforward neural network with even a single hidden layer, given enough neurons and an appropriate activation function, can approximate any continuous function on a compact domain to any desired degree of accuracy. This is a remarkable result: it means the limitation of neural networks is not one of theoretical expressive power, but rather of practical considerations such as network size, training data, and optimization.

The goal of this paper is to build up the mathematical background necessary to state and prove one of the foundational versions of this theorem, due to Cybenko (1989), and to walk through its proof in detail. Along the way, we will introduce the core ideas from functional analysis, topology, and measure theory that make the proof possible, including metric spaces, open and closed sets, dense sets, measures, linear functionals, and discriminatory functions. These concepts may seem far removed from the practical world of neural networks, but they are precisely what allow us to make rigorous the intuitive idea that neural networks are "universal approximators."

By the end of this paper, the reader should understand not only what the Universal Approximation Theorem says, but why it is true, and appreciate the elegant interplay between abstract mathematics and one of the most widely used tools in modern computing.

3 History

3.1 The preceptron

At its heart, the perceptron was elegantly simple: Inputs: Think of them as sensors — pixels in an image, for example. Weights: Each input had a weight, representing its importance. Summation: The inputs were multiplied by weights and summed. Activation: If the sum crossed a threshold, the perceptron "fired" — producing an output of 1 instead of 0. This mimicked how neurons in the brain process signals. The magic, however, was in learning. Rosenblatt introduced a rule to adjust weights based on mistakes: if the perceptron got the answer wrong, tweak the weights to reduce future error. Over time, the perceptron could be trained to classify patterns — letters, shapes, even simple images. In modern terms, Rosenblatt had invented the first trainable neural network.

3.2 The mark I preceptron

Rosenblatt's vision was not just theoretical. In 1960, he and his team built the Mark I Perceptron, a physical machine housed in a cabinet of wires, motors, and electronic components. It had an array of photocells for input, random initial weights (set with motors), and adaptive mechanisms to update them during training. The perceptron could learn to recognize simple images — for example, distinguishing between triangles and squares. It was clunky by today's standards, but for 1960, it was revolutionary. A machine that could learn from experience, rather than being explicitly programmed, seemed almost alive.

3.3 The Hype

Rosenblatt himself believed in the perceptron's potential. He predicted that, given enough layers and units, perceptrons could eventually learn to translate languages, recognize speech, and even achieve general intelligence. The press eagerly amplified these claims. The New York Times declared: "The Navy revealed the embryo of an electronic computer today that it expects will be able to walk, talk, see, write, reproduce itself and be conscious of its existence." Though Rosenblatt himself was more cautious than these headlines suggest, the hype was irresistible. Neural networks became a hot topic, attracting funding from the U.S. military and academic interest across disciplines.

3.4 Backlash

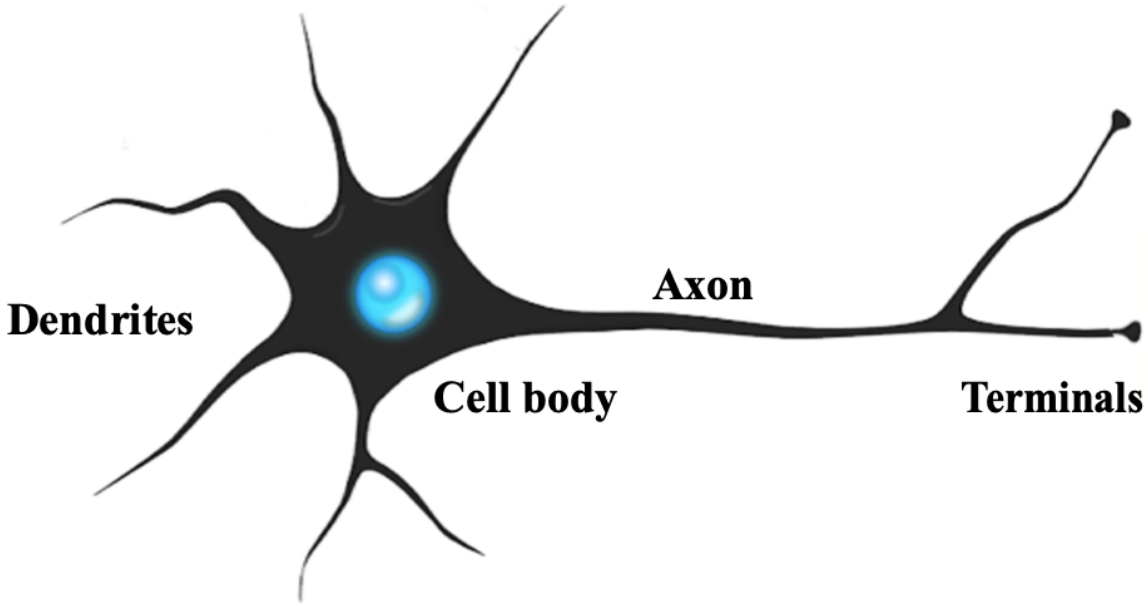
In 1969, Marvin Minsky and Seymour Papert published Perceptrons, a critical mathematical analysis of Rosenblatt's model. They proved that single-layer perceptrons had severe limitations. Most famously, they could not solve the XOR problem — a simple logical function that requires combining inputs in a non-linear way. While Minsky and Papert acknowledged that multi-layer networks could overcome this, the training methods for such networks were not yet known. Their critique, though technical, was interpreted broadly as a death sentence for neural networks. Funding dried up. AI research shifted toward symbolic reasoning and expert systems. The first AI winter had begun. Rosenblatt, sadly, did not

live to defend or extend his work. He died tragically young in a boating accident in 1971, at just 43. His dream of intelligent learning machines seemed to fade with him.

3.5 Rediscovery

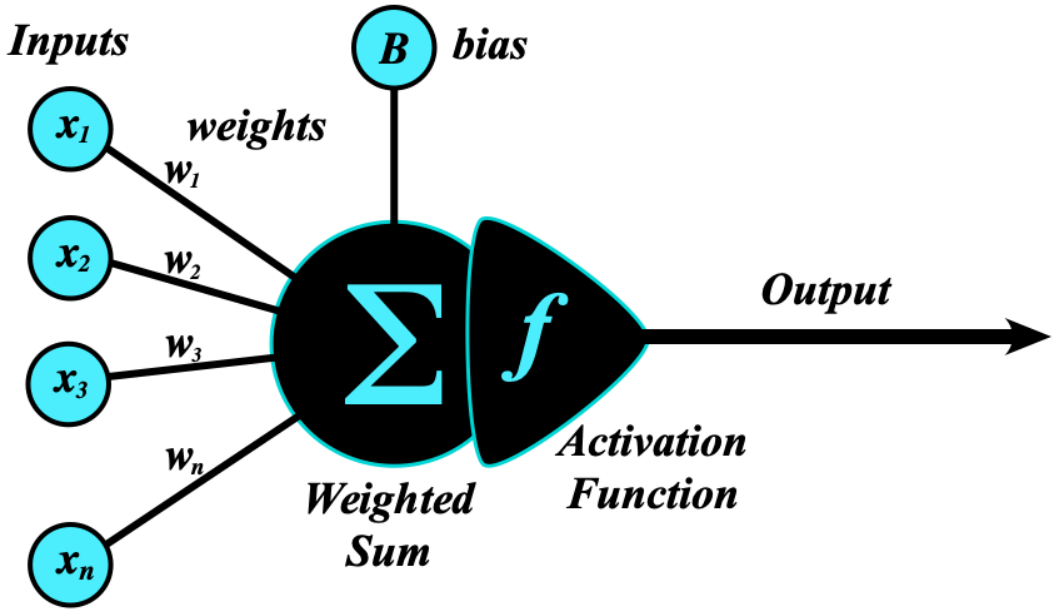
In the 1980s, researchers rediscovered the power of multi-layer perceptrons combined with the backpropagation algorithm, which allowed networks to adjust weights across multiple layers. Suddenly, the very limitations that Minsky and Papert highlighted could be overcome. By the 2010s, with faster hardware and massive datasets, neural networks exploded back into the spotlight. Deep learning, essentially stacked perceptrons with nonlinear activations, began to dominate computer vision, speech recognition, and natural language processing. Every modern neural network, from convolutional nets to transformers, is a direct descendant of Rosenblatt's perceptron.

3.6 Modelling the human brain



Neurons receive signals through branch-like dendrites, process them in the cell body, and send electrical impulses down a long tail called an axon. Once the signal reaches the end, it releases chemical messengers (neurotransmitters) across a gap called a synapse to excite or inhibit the receiving neuron.
Neurons are the main building blocks of the brain. A brain contains almost 86 billion neurons. Together, they form an intricate network with over 100 trillion synaptic connections.

Artificial neurons follow the same enormous structure, too. Llama 4 Scout, for example, has 109 billion total parameters (the equivelent of neurons).

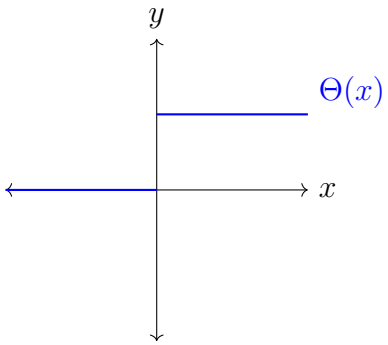


Artificial neurons, similarly, process data by taking inputs, multiplying them by specific weights, adding a bias, and passing the result through an activation function to determine the final output. We now delve more into neural networks and how they work.

Inputs x through x_n, representing our data, are multiplied by weights. You could think of weights as indicators of how important a specific input, also called feature, is.

We then sum all these values; add a bias, a scalar number; then pass the resulting number through an activation function and the number is outputed.

This activation function models how biological neurons work, either firing or not. This is represnted as a step function which outputs either zero if the number is negative or zero, or 1 if it is positive.



step function

Lets now apply neural networks to a real problem. This one isi called the OR problem. You want a functin that outputs takes in these cordinates and outputs the 0 or 1.

$$\begin{array}{c|c} (0,\,0) & 0 \\ (0,\,1) & 1 \\ (1,\,0) & 1 \\ (1,\,1) & 1 \end{array} \tag{3.1}$$

The model assigns $w_1 = 1, w_2 = 1, b_1 = -.5$
We can express our equation as $f_{NN}(x_1, x_2) = w_1 x_1 + w_2 x_2 + b_1$

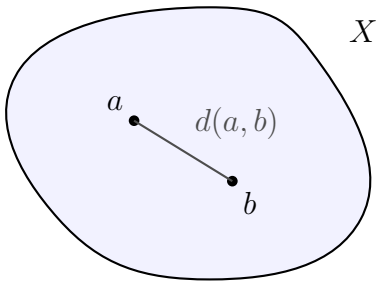
$$\begin{aligned} \text{For } (0,0): &\quad 1(0) + 1(0) - 0.5 = -0.5 \to 0 \\ \text{For } (1,0): &\quad 1(1) + 1(0) - 0.5 = 0.5 \to 1 \\ \text{For } (0,1): &\quad 1(0) + 1(1) - 0.5 = 0.5 \to 1 \\ \text{For } (1,1): &\quad 1(1) + 1(1) - 0.5 = 1.5 \to 1 \end{aligned}$$

This neural network learned this comopletely on its own. The process was completley automatic. The model uses an algorithm to minimize the loss function (the difference between the predicted output and the correct output) to automatically optimize the weights and biases. This is the simplest problem an NN could solve; There are much more complex problems. classification, for example, like deciding between cats and dogs, though they work with the foundations discussed above, NNs have became much more complex as time went on.

4 Preliminaries

Machine learning theory is a complex field that sits at the intersection of many fields such as real analysis, topology, measure theory, and functional analysis. This section lays out the concepts needed for a better understanding of the UATs.

4.1 Metric Spaces

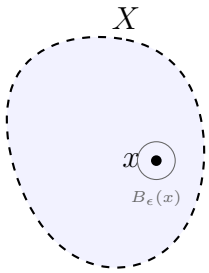


Definition 1.1 (Metric Space)**.** *A metric space is a set X together with a function $d : X \times X \to \mathbb{R}$ such that $d(x_1, x_2) \geq 0$ for all $x, y \in X$, and $d(x, y) = 0$ iff $x = y$.*

Imagine a set X, a and b are two elements of set X, if we define a function d that takes in 2 elements of the set and returns a number, a measure of the distance between them, we then have a metric. The choice of how the metric calculates the distance is up to us to

choose for a specific set. So our function is defined as $d(x_1, x_2) = \sqrt{x_1^2 - y_1^2}$ Now you have a metric space! A space X with metric d that can be expressed as an object (X, d).
There are certain rules metric spaces must abide by but they are not much of our concern. Formally it is:

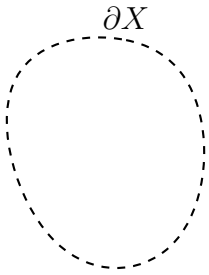
4.2 Open Sets



Open Set X

$A \subseteq X$ is called **open** if for each $x \in A$ there is an open ball with $\mathcal{B}_\varepsilon(x) \subseteq A$.

4.3 Boundary Points



Boundary ∂X

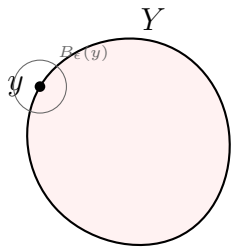
$A \subseteq X$. $x \in X$ is called a **boundary point for** A if for all $\varepsilon > 0$:

$$\mathcal{B}_\varepsilon(x) \cap A \neq \emptyset \quad \text{and} \quad \mathcal{B}_\varepsilon(x) \cap A^c \neq \emptyset \quad [A^c := X \setminus A]$$

$$\partial A := \{x \in X \mid x \text{ is boundary point for } A\}$$

Remember: A open $\iff$ $A \cap \partial A = \emptyset$

4.4 Closed sets

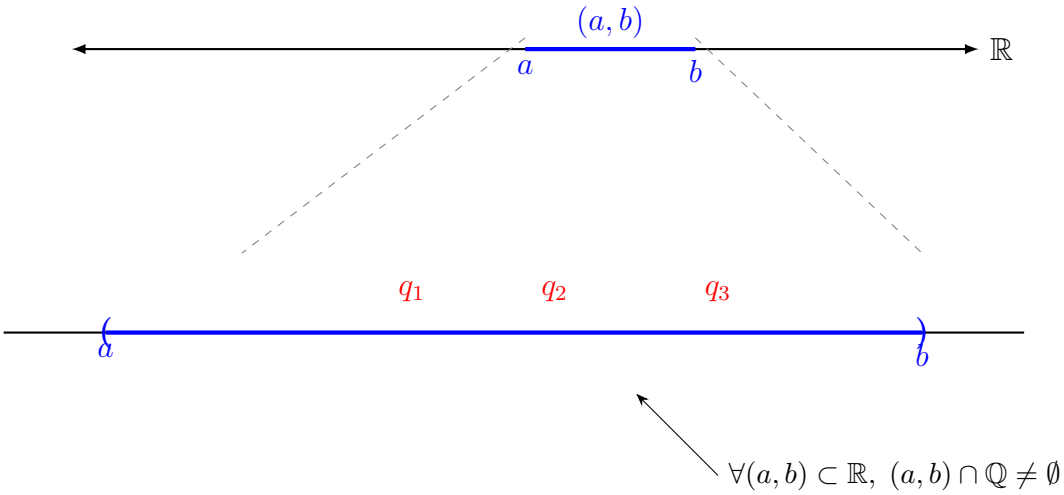


Closed Set Y

$A \subseteq X$ is called **closed** if $A^c := X \setminus A$ is open.
Remember: A closed $\iff \partial A \subseteq A$

To be as simple as possible, an open set is one that does not contain it's boundary points, while a closed set is one that does.

4.5 Dense Set



A subset A of a topological space (X, τ) is said to be **dense** in X if its closure equals X:

$$\overline{A} = X$$

Alternatively, in a metric space (X, d), A is dense in X if for every $x \in X$ and every $\epsilon > 0$, there exists an $a \in A$ such that $d(x, a) < \epsilon$.

Measures

Given a measurable space (X, Σ), a **measure** μ is a function $\mu : \Sigma \to [0, \infty]$ that satisfies:

1. $\mu(A) \geq 0$ for all $A \in \Sigma$
2. $\mu(\emptyset) = 0$

3. For all sequences of pairwise disjoint sets $\{A_n\}_{n=1}^{\infty} \subseteq \Sigma$:

$$\mu\left(\bigcup_{n=1}^{\infty} A_n\right) = \sum_{n=1}^{\infty} \mu(A_n)$$

6. Functionals

A **functional** is a mapping $f : V \to F$ from a vector space V into its underlying scalar field F (typically $\mathbb{R}$ or $\mathbb{C}$). It is a linear functional if for all $x, y \in V$ and $\alpha, \beta \in F$:

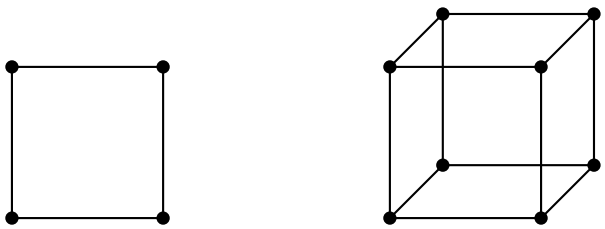
$$f(\alpha x + \beta y) = \alpha f(x) + \beta f(y)$$

7. Hypercube

An n-dimensional unit **hypercube** I^n in $\mathbb{R}^n$ is the Cartesian product of n closed unit intervals:

$$I^n = [0,1]^n = \underbrace{[0,1] \times [0,1] \times \cdots \times [0,1]}_{n \text{ times}}$$

0-Cube (Point) 1-Cube (Line Segment)



2-Cube (Square) 3-Cube (Cube)

8. Discriminatory Functions

A Borel measurable function $\sigma : \mathbb{R} \to \mathbb{R}$ is said to be **discriminatory** if for any signed finite Borel measure μ on I^n, the condition:

$$\int_{I^n} \sigma(w \cdot x + b)\, d\mu(x) = 0$$

for all $w \in \mathbb{R}^n$ and $b \in \mathbb{R}$ implies that $\mu = 0$.

5 Main Results

Let I_n denote the n-dimensional unit cube, $[0,1]^n$. The space of continuous functions on I_n is denoted by $C(I_n)$ and we use $\|f\|$ to denote the supremum (or uniform) norm of an $f \in C(I_n)$. In general we use $\|\cdot\|$ to denote the maximum of a function on its domain. The space of finite, signed regular Borel measures on I_n is denoted by $M(I_n)$. See [R2] for a presentation of these and other functional analysis constructions that we use.

The main goal of this paper is to investigate conditions under which sums of the form

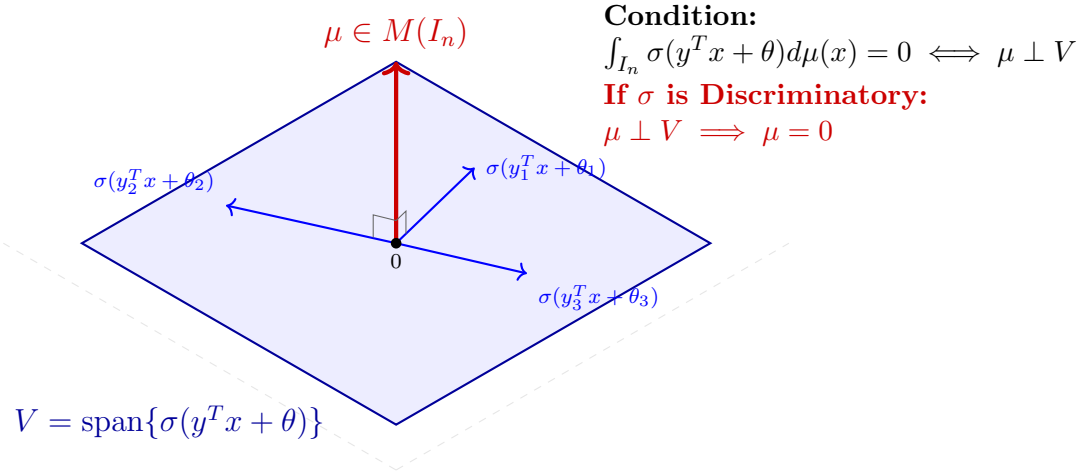
$$G(x) = \sum_{j=1}^{N} \alpha_j \sigma(y_j^T x + \theta_j)$$

are dense in $C(I_n)$ with respect to the supremum norm.

Definition. We say that σ is *discriminatory* if for a measure $\mu \in M(I_n)$

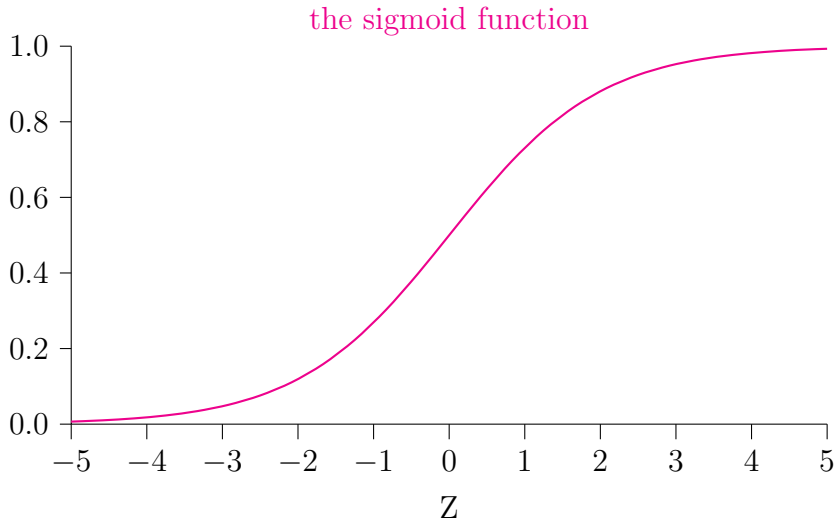
$$\int_{I_n} \sigma(y^T x + \theta)\, d\mu(x) = 0$$

for all $y \in \mathbb{R}^n$ and $\theta \in \mathbb{R}$ implies that μ itself must be 0 inn any other case.



Definition. We say that σ is *sigmoidal* if

$$\sigma(t) \to \begin{cases} 1 & \text{as } t \to +\infty, \\ 0 & \text{as } t \to -\infty. \end{cases}$$

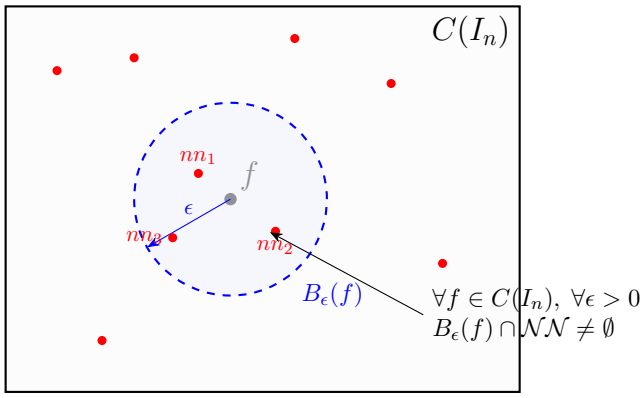


Theorem 1. *Let σ be any continuous discriminatory function. Then finite sums of the form*

$$G(x) = \sum_{j=1}^{N} \alpha_j \sigma(y_j^T x + \theta_j) \tag{5.1}$$

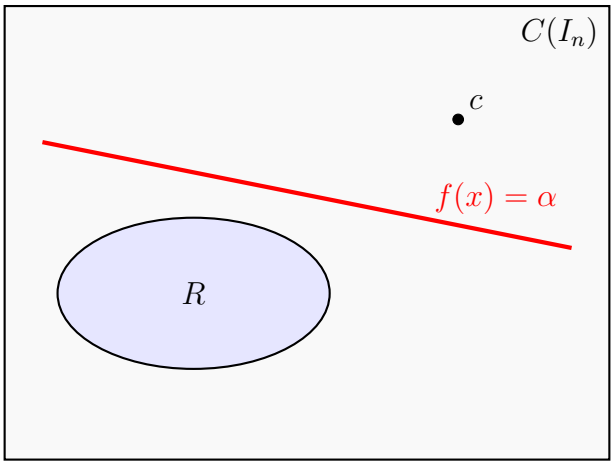
are dense in $C(I_n)$. In other words, given any $f \in C(I_n)$ and $\varepsilon > 0$, there is a sum, $G(x)$, of the above form, for which

$$|G(x) - f(x)| < \varepsilon \quad \textit{for all } x \in I_n.$$



Proof

Let $S \subset C(I_n)$ of all possible functions of the form $G(x)$ as in 2. We claim that the closure in S (R) is dense in $C(I_n)$. Assume that R is not dense in $C(I_n)$, this means that R is a proper closed subspace in $C(I_n)$. By the Hahn-Banach theorem, there must exist a linear functional, say L, such that $L \neq 0$ (L is not a zero functional), and that $L(S) = L(R) = 0$.



By the Riesz Representation theorem, this functional is in the form

$$L(h) = \int_{I_n} h(x)\, d\mu(x)$$

for some $\mu \in M(I_n)$, and all $h \in C(I_n)$. In particular, since $\sigma(y^T x + \theta)$ is in R for all y and θ, we must have that

$$\int_{I_n} \sigma(y^T x + \theta)\, d\mu(x) = 0$$

However, we assumed that σ was discriminatory, so that this condition implies that $\mu = 0$ contridictinig our assumption that S is not dense in $C(I_n)$, therefore S must be dense in $C(I_n)$.

This demonstrates that sums of the form

$$G(x) = \sum_{j=1}^{N} \alpha_j\, \sigma\big(y_j^T x + \theta_j\big)$$

are dense in $C(I_n)$ providing that σ is continuous and discriminatory.

Now, we specialize this result to show that any continuous sigmoidal σ of the form discussed before, namely

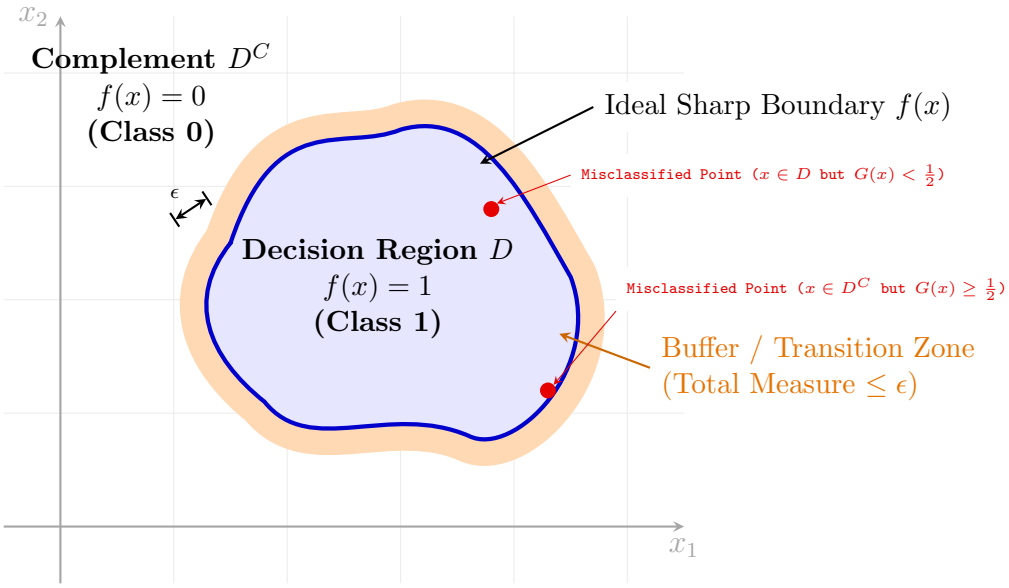
$$\sigma(t) \to \begin{cases} 1, & \text{as } t \to +\infty, \\ 0, & \text{as } t \to -\infty, \end{cases}$$

is discriminatory. It is worth noting that, in neural network applications, continuous sigmoidal activation functions are typically taken to be monotonically increasing, but no monotonicity is required in our results.

Definition 1.2 (Lusin's Theorem)**.** *A theorem in real analysis stating that any discontinuous function can be treated as a perfectly smooth, continuous function if restricted to a domain that excludes a tiny error region of arbitrary size ϵ.*

Definition 1.3 (Universal Approximation (Theorem 2))**.** *An established mathematical result proving that a single-hidden-layer neural network can approximate any strictly continuous function to any desired degree of accuracy.*

Theorem 1. Let σ be a continuous sigmoidal function, and let $f(x)$ be a sharp, discontinuous binary decision function on a domain I_n. For any chosen error threshold $\epsilon > 0$:



1. There exists a single-hidden-layer neural network:

$$G(x) = \sum_{j=1}^{N} \alpha_j \sigma(y_j^T x + \theta_j)$$

2. There exists a valid subset $D \subset I_n$ whose total size satisfies:

$$m(D) \geq 1 - \epsilon$$

3. The network approximates the decision function closely within this valid subset:

$$|G(x) - f(x)| < \epsilon \quad \text{for all } x \in D$$

Proof. The proof bridges the gap between the discontinuous target function $f(x)$ and the smooth neural network $G(x)$ via an intermediate continuous function $h(x)$:

1. **Smooth the Target:** By Lusin's Theorem, choose a perfectly continuous function $h(x)$ and a valid set D (where the excluded error zone has a maximum size of ϵ) such that the continuous function matches the target function identically:

$$h(x) = f(x) \quad \text{for all } x \in D$$

2. **Approximate the Continuous Function:** Because $h(x)$ is continuous, Theorem 2 applies. This guarantees the existence of a neural network $G(x)$ that approximates $h(x)$ within an accuracy threshold of ϵ over the entire domain:

$$|G(x) - h(x)| < \epsilon$$

3. **Combine Terms:** For any point x belonging to the valid set D, substitute $f(x)$ in place of $h(x)$:

$$|G(x) - f(x)| = |G(x) - h(x)| < \epsilon$$

Thus, the neural network $G(x)$ successfully approximates the discontinuous decision function $f(x)$ everywhere except on a negligible set of maximum size ϵ. ■

6 A cknowledgements

I would like to thank Dr. Simon for the euler circle program. And my TA, Aditya, for helping me through the process of writing the paper and understanding the concetps introduced here.

7 Bibliography

1. Cybenko, G. (1989). Approximation by superpositions of a sigmoidal function. Mathematics of Control, Signals and Systems, 2(4), 303–314.

2. Augustine, M. T. (2024). A survey on universal approximation theorems. arXiv preprint arXiv:2407.12895v1.