Isotonic Regression Using Positive Neural Networks

Idriss Benkirane
CentraleSupélec – PSL, Paris
idriss.benkirane@student-cs.fr

Abstract

Monotone operators play a central role in the study of inverse problems, owing to their structural properties that facilitate analysis and resolution. Such operators arise naturally in various contexts, including physics, economics, and machine learning, where they model relationships between variables of interest. Our main objective is to design a method that, given a corrupted operator T, recovers a monotone operator $\widetilde{T}$ that is "sufficiently close" to T. The notion of proximity will be defined according to criteria tailored to the nature of the problem and the constraints of the intended application.

1 Introduction

The classical approach to regressing a monotone operator is to constrain its spectrum on a specific dataset. After training, the operator is guaranteed to be monotone only within the portion of the space observed during training. In other words, monotonicity outside the training distribution is not guaranteed.

For example, if we train our model on images of dogs, nothing ensures that the learned operator $\widetilde{T}$ will behave monotonically when applied to images of cats. This limitation can be problematic, especially in dynamic settings where the data distribution may evolve.

Our goal is to develop a framework that guarantees monotonicity of $\widetilde{T}$ regardless of the underlying data distribution. To build intuition, we first examine the one-dimensional setting.

2 Isotonic Regression in 1D

2.1 Historical Overview

Problem 2.1 (Isotonic Median Regression [**?**])**.** *Let $S_1, \ldots, S_n$ be sets of real numbers, with $S_i = \{y_{i,j} : 1 \leq j \leq m_i\}$ for $i = 1, \ldots, n$. The isotonic median regression problem with respect to a complete order is*

$$\textit{(IMR)} \quad \begin{cases} \min \sum_{i=1}^{n} \sum_{j=1}^{m_i} |y_{i,j} - x_i| \\ \textit{subject to } x_1 \leq x_2 \leq \cdots \leq x_n. \end{cases}$$

The contribution of Chakravarti's paper was to provide a dual formulation of this problem that can be solved using linear programming techniques.

2.2 The PAV Algorithm

The Pool Adjacent Violators (PAV) algorithm offers an efficient approach for solving isotonic regression problems. The intuition is that if S_i and S_j are disjoint sets such that $M(S_i) \leq M(S_j)$, then

$$M(S_i) \leq M(S_i \cup S_j) \leq M(S_j),$$

where $M(S)$ denotes the median of set S.

Algorithm 1 Pool Adjacent Violators (PAV) Algorithm

```
Initialize J ← {{1}, {2}, ..., {n}}, B_0 ← {1}
while B_+ ≠ ∅ do
    if M(B_0) ≤ M(B_+) then
        B_0 ← B_+
    else
        Merge: J ← J \ {B_0, B_+} ∪ (B_0 ∪ B_+)
        B_0 ← B_0 ∪ B_+
        while B_− ≠ ∅ and M(B_−) > M(B_0) do
            Merge: J ← J \ {B_0, B_−} ∪ (B_0 ∪ B_−)
            B_0 ← B_0 ∪ B_−
        end while
    end if
end while
return current partition J (optimal solution)
```

2.3 Nearly-Isotonic Regression

(author?) [1] proposed a relaxation of the isotonic constraint:

Problem 2.2 (Nearly-Isotonic Regression)**.** *Given $y_1, \ldots, y_n$, solve*

$$\hat{\beta}_\lambda = \arg\min_{\beta \in \mathbb{R}^n} \frac{1}{2} \sum_{i=1}^{n} (y_i - \beta_i)^2 + \lambda \sum_{i=1}^{n-1} (\beta_i - \beta_{i+1})_+,$$

where $x_+ = x \cdot \mathbf{1}(x > 0)$ and $\lambda \geq 0$.

As noted, the approach of the PAV algorithm is quite "greedy". This motivates us to investigate whether it is possible to outperform the PAV algorithm.

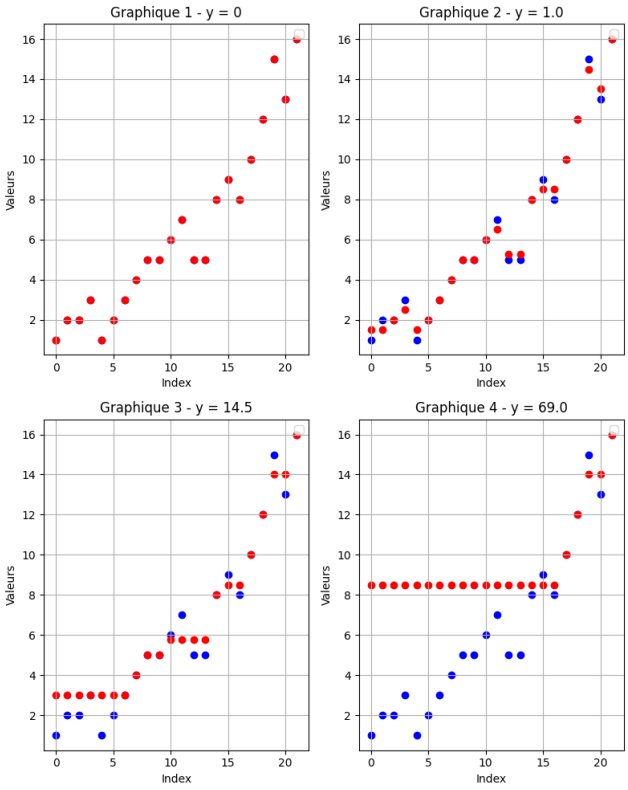


Figure 1: Example of Mpav: blue points denote the observed data, red points the regression estimate.

2.4 Projection Formulation

Problem 2.3 (Projection onto the Cone of Isotonic Functions)**.** *Given an observation function $y : [-1, 1] \to \mathbb{R}$, we seek an increasing function $f \in C$ that minimizes the squared L^2 discrepancy*

$$\min_{f \in C} \int_{-1}^{1} \big(y(x) - f(x)\big)^2 \, dx,$$

where C denotes the convex cone of isotonic (non-decreasing) functions on $[-1, 1]$.

Proposition 2.4. *The solution to Problem 2.3 is the orthogonal projection $P_C(y)$ of y onto C, characterized by*

$$\langle y - P_C(y),\, g - P_C(y) \rangle \leq 0 \quad \forall g \in C.$$

Therefore, our objective is to identify the increasing function $P_C(y)$ that best approximates y. Even a good approximation of $P_C(y)$, without exact recovery, represents a valuable step forward.

Since $L^2([-1, 1])$ is a separable Hilbert space, one may consider classical bases such as trigonometric functions (Fourier basis) or polynomials (Weierstrass theorem). Our idea is instead to construct a *pseudo-basis* of increasing functions that spans the convex cone C. This would allow us to reconstruct $P_C(y)$ in a manner analogous to Fourier expansions.

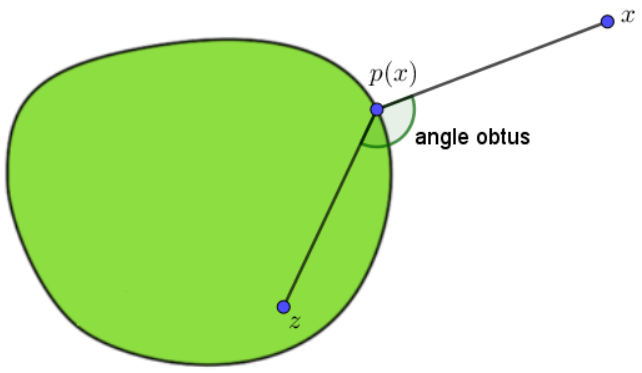


Figure 2: Projection onto a convex set.

3 Universal Approximation of Neural Networks

The emergence of neural networks in the 1980s revealed a new basis for approximating measurable and continuous functions. We denote by $\mathcal{C}(X, Y)$ the set of continuous functions from X to Y. The classical form of the universal approximation theorem for arbitrary width and depth 1 is:

Theorem 3.1 (Universal Approximation)**.** *Let $\sigma \in C(\mathbb{R}, \mathbb{R})$. Then σ is non-polynomial* if and only if *for all $n, m \in \mathbb{N}$, every compact $K \subseteq \mathbb{R}^n$, every $f \in C(K, \mathbb{R}^m)$, and every $\epsilon > 0$, there exist $k \in \mathbb{N}$, $A \in \mathbb{R}^{k \times n}$, $b \in \mathbb{R}^k$, and $C \in \mathbb{R}^{m \times k}$ such that*

$$\sup_{x \in K} \|f(x) - g(x)\| < \epsilon,$$

where

$$g(x) = C \cdot \sigma(Ax + b).$$

The number of parameters required depends on the regularity of f in the Sobolev sense. Barron (1991) further showed that this number may grow exponentially with $1/\epsilon$. Deeper architectures, however, can reduce the number of required parameters, as shown by Bach in the context of radial functions.

3.1 Positive Neural Networks

In the design of neural networks, precision alone does not fully capture model quality. Robustness often characterized via the Lipschitz constant plays a crucial role in evaluating the stability and generalization of learned models.

Definition 3.2 (Lipschitz Constant)**.** *The* Lipschitz constant *of a function $f : \mathbb{R}^d \to \mathbb{R}^m$ is the smallest $L \geq 0$ such that*

$$\|f(x_1) - f(x_2)\| \leq L \|x_1 - x_2\|, \quad \forall x_1, x_2 \in \mathbb{R}^d.$$

The Lipschitz constant quantifies the sensitivity of the model to input perturbations and provides a refined measure of generalization ability.

Networks with positive weights are generally more robust, with Lipschitz constant bounded by $\|W_1 \cdots W_n\|_s$. A natural question is whether such networks preserve the universal approximation property.

Unfortunately, no. When the activation functions are increasing, a positive neural network necessarily preserves order and therefore cannot approximate a decreasing function, for example.

3.2 Adherence of Positive Neural Networks

Rather than attempting to approximate arbitrary functions, we now ask whether positive neural networks can approximate the subclass of *increasing* functions.

Conjecture 3.3 (Positive Neural Network Approximation)**.** *Let $K \subset \mathbb{R}$ be compact, $f \in C(\mathbb{R})$ an increasing function, $\epsilon > 0$, and $\sigma \in C(\mathbb{R})$ an increasing activation function. Does there exist $n \in \mathbb{N}$ and parameters $W_1 \in M_{n,1}(\mathbb{R}_+)$, $W_2 \in M_{1,n}(\mathbb{R}_+)$, $b \in \mathbb{R}^n$, and $C \in \mathbb{R}$ such that*

$$\sup_{x \in K} \left| f(x) - \Big(W_2 \cdot \sigma(W_1 x + b) + C \Big) \right| \leq \epsilon \, ?$$

Approximation of Increasing Convex Functions. This class of functions exhibits an *acceleration* phenomenon.

Objective 3.4 (ReLU Approximation of Convex Increasing Functions)**.** *Let $f : \mathbb{R} \to \mathbb{R}$ be continuous, increasing, and convex. Let $K = [a, b] \subset \mathbb{R}$ be compact, and fix $\epsilon > 0$. The goal is to construct a positive neural network F with ReLU activation such that*

$$\|f - F\|_{\infty, K} < \epsilon.$$

Construction. Since f is continuous on the compact set K, Heine's theorem implies that f is uniformly continuous on K. Hence,

$$\exists \, \delta > 0 \quad \text{such that} \quad \forall x, y \in K, \ |x - y| < \delta \implies |f(x) - f(y)| < \epsilon.$$

Because K is compact and connected, in one dimension we may write $K = [a, b]$ with $a, b \in \mathbb{R}$. We now consider a uniform subdivision of $[a, b]$ with step size δ:

$$a = x_0 < x_1 < \cdots < x_n = b, \quad x_i = a + i\delta, \ i = 0, 1, \ldots, n.$$

The idea is to approximate f by affine functions whose slopes are increasing, a direct consequence of convexity.

The function g interpolates f at the points x_i. By uniform continuity, we deduce

$$\|f - g\|_{\infty, K} < \epsilon.$$

Neural Network Construction. We now construct a positive neural network that exactly reproduces g. Define

$$F(x) = I \cdot \mathrm{ReLU}(W \cdot x + b) + c,$$

with $W \in \mathcal{M}_{n,1}$, $I \in \mathcal{M}_{1,n}$, $b \in \mathbb{R}^n$, and $c \in \mathbb{R}$. Equivalently,

$$F(x) = (w_1 x + b_1)_+ + \cdots + (w_n x + b_n)_+ + c.$$

Here $n = (b - a)/\delta$ is the number of neurons in the hidden layer.

The idea is to activate $(w_i x + b_i)_+$ only on $[x_i, b]$ (at each interval $[x_i, x_{i+1}]$ a new ReLU is activated). This yields the following rule:

$$w_{i+1} x_i + b_{i+1} = 0 \quad \text{(to enforce nullity on the left)},$$

and

$$w_1 + \cdots + w_{i+1} = \mathrm{slope}\big(g|_{[x_i, x_{i+1}]}\big).$$

Since the convexity of f ensures that the sequence of slopes $(\mathrm{slope}(g|_{[x_i, x_{i+1}]}))_i$ is increasing, we deduce that $w \geq 0$. Therefore $g = F$ on K and

$$\|f - F\|_{\infty, K} < \epsilon.$$

Conclusion. Any increasing convex function can be approximated arbitrarily well by a ReLU network with positive weights. However, such architectures are limited to convex functions. Indeed, a sum of the form

$$a_1(w_1 x + b_1)_+ + \cdots + a_n(w_n x + b_n)_+ + c, \quad a_i \geq 0,$$

is itself convex, so ReLU networks with positive weights cannot approximate arbitrary increasing functions.

Thus, in order to treat the general case of increasing functions, the ReLU activation cannot work directly.

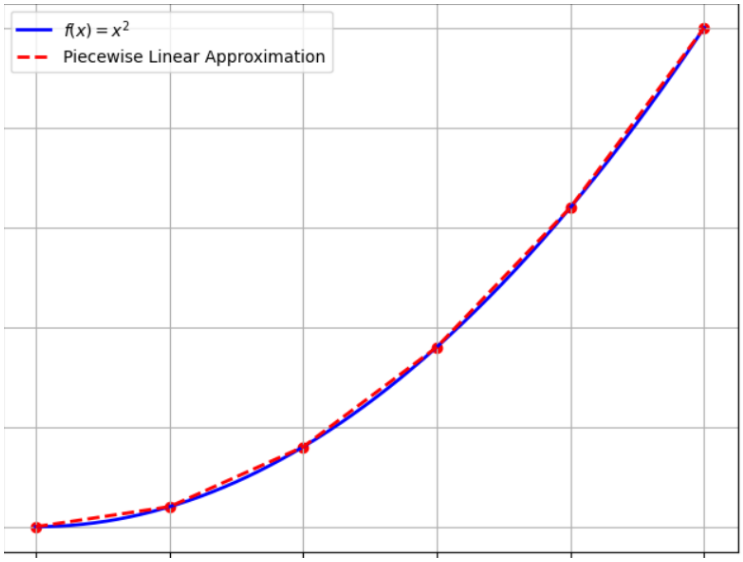


Figure 3: Approximation

General Case. Two natural extensions suggest themselves:

- **Adding a layer:** a deeper network could approximate functions with both convex and concave segments.
- **Changing the activation:** using activation functions that exhibit both convexity and concavity, such as sigmoids, offers greater flexibility.

3.3 Sigmoid Activations

The sigmoid activation function is convex for $x \leq 0$ and concave for $x \geq 0$, making it a promising candidate for approximating general increasing functions.

Theorem 3.5 (Approximation with Sigmoids)**.** *Let $K \subset \mathbb{R}_+$ be compact, $f \in C(\mathbb{R})$ increasing, and σ a sigmoid activation function. Then for every $\epsilon > 0$ there exist $W_1 \in M_{n,1}(\mathbb{R}_+)$, $W_2 \in M_{1,n}(\mathbb{R}_+)$, $b \in \mathbb{R}^n$, and $C \in \mathbb{R}$ such that*

$$\sup_{x \in K} |f(x) - (W_2 \cdot \sigma(W_1 x + b) + C)| \leq \epsilon.$$

The fundamental concept of the prove is the ability of a sigmoid-activated neural networks to approximate the Heaviside step function. By adjusting the weight parameter w in the function $\sigma(W_1 \cdot x + b)$, the sigmoid's transition can be made arbitrarily steep, closely resembling the Heaviside step. This allows for the construction of piecewise constant approximations, and through combinations of such approximations, more complex continuous functions can be approximated on compact sets.

Remark. For all $x \in \mathbb{R}$, the sigmoid function is given by

$$\sigma(x) = \frac{\exp(x)}{1 + \exp(x)}.$$

Let $h(x) = \sigma(wx + b)$. Then its derivative is

$$h'(x) = \frac{w \cdot \exp(wx + b)}{\big(1 + \exp(wx + b)\big)^2}.$$

Hence, the abrupt change in the value of $\sigma(wx + b)$ occurs near

$$t = -\frac{b}{w}.$$

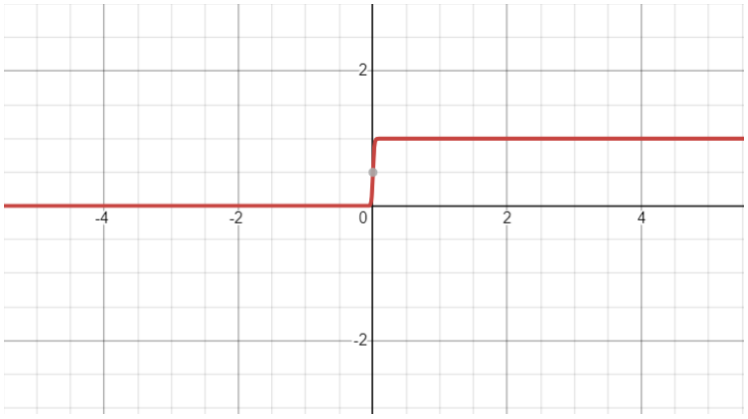


Figure 4: avec $w = 100, b = 0$

Proof. Since f is continuous on the compact set $K = [a, b]$, there exists a step function E such that

$$\|f - E\|_{\infty,K} < \epsilon.$$

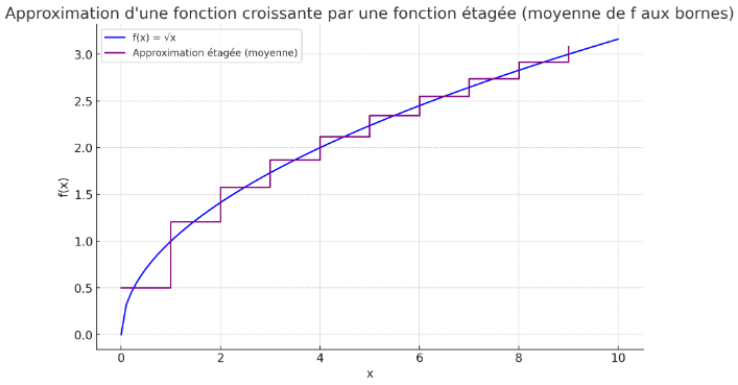


Figure 5: Approximation by a step function

Because E is a step function on $[a, b]$, there exists a partition

$$a = x_0 < x_1 < \cdots < x_N = b$$

such that E is constant on each interval $[x_i, x_{i+1}]$.

We approximate E by a positive neural network of the form

$$F(x) = \sum_{i=1}^{N} a_i \, \sigma(wx + b_i) + c.$$

The discontinuity points are chosen such that

$$x_i = -\frac{b_i}{w}, \quad i = 1, \ldots, N.$$

To simplify notation, we may impose $w_1 = w_2 = \cdots = w_N = w$ with $w \gg 1$.

On the interval $[x_0, x_1]$. We set

$$a_1 = E|_{[x_1,x_2]} - E|_{[x_0,x_1]}, \quad b_1 = -x_1 w, \quad c = E|_{[x_0,x_1]}.$$

On the interval $[x_1, x_2]$. We take

$$a_2 = E|_{[x_2,x_3]} - E|_{[x_1,x_2]}, \quad b_2 = -x_2 w,$$

with no additional bias term.

By induction. For general i, we define

$$a_i = E|_{[x_i, x_{i+1}]} - E|_{[x_{i-1}, x_i]}, \quad b_i = -x_i w, \quad w \gg 1.$$

With this construction, F approximates E, and therefore, by transitivity,

$$\|f - F\|_{\infty, K} < 2\epsilon.$$

Remark. Since f is increasing, the step function E is also increasing. Thus, the coefficients a_i are always nonnegative. This property is essential to ensure that the constructed neural network is *positive*.

If smoother approximations are required, an additional convolutional layer can be appended.

Having established the one-dimensional case, we now turn to higher dimensions.

4 Isotonic Regression from $\mathbb{R}^2$ to $\mathbb{R}$

We now extend the one-dimensional study to higher dimensions. We begin with the case $d = 2$. The main difficulty is that the order relation on $\mathbb{R}^2$ is only partial.

Definition 4.1 (Isotonic Function in $\mathbb{R}^2$)**.** *A function $f : \mathbb{R}^2 \to \mathbb{R}$ is called* isotonic *if, for all $(x_1, y_1), (x_2, y_2) \in \mathbb{R}^2$,*

$$x_1 \leq x_2 \text{ and } y_1 \leq y_2 \implies f(x_1, y_1) \leq f(x_2, y_2).$$

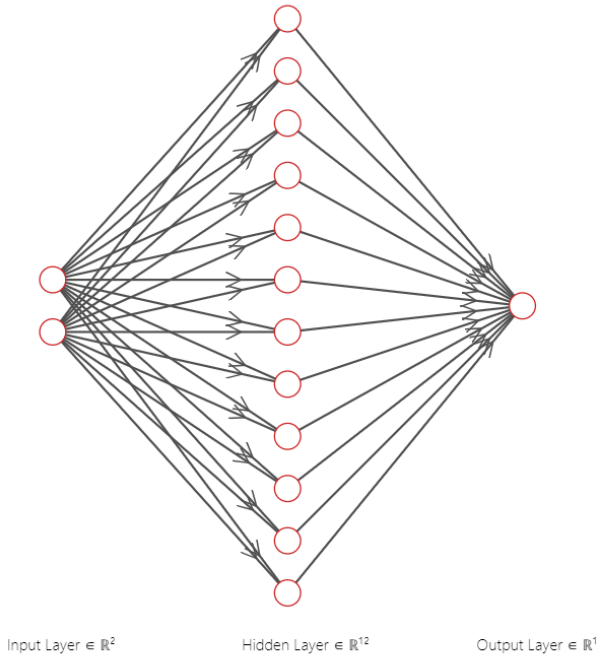


Figure 6: neural network with one hidden layer

Objective. Our goal is to design a neural network architecture that is:

1. *guaranteed* to be isotonic,
2. sufficiently expressive to approximate any isotonic function.

4.1 Step Functions and Density

Proposition 4.2. *The set of step functions from $\mathbb{R}^2$ to $\mathbb{R}$ is dense in the set of isotonic functions.*

Sketch. Since isotonic functions are measurable, they can be approximated arbitrarily well by step functions adapted to a uniform grid. □

Thus, to approximate isotonic functions, it suffices to construct a neural network capable of generating isotonic step functions.

4.2 Sigmoid-Based Construction

Consider a single-hidden-layer neural network:

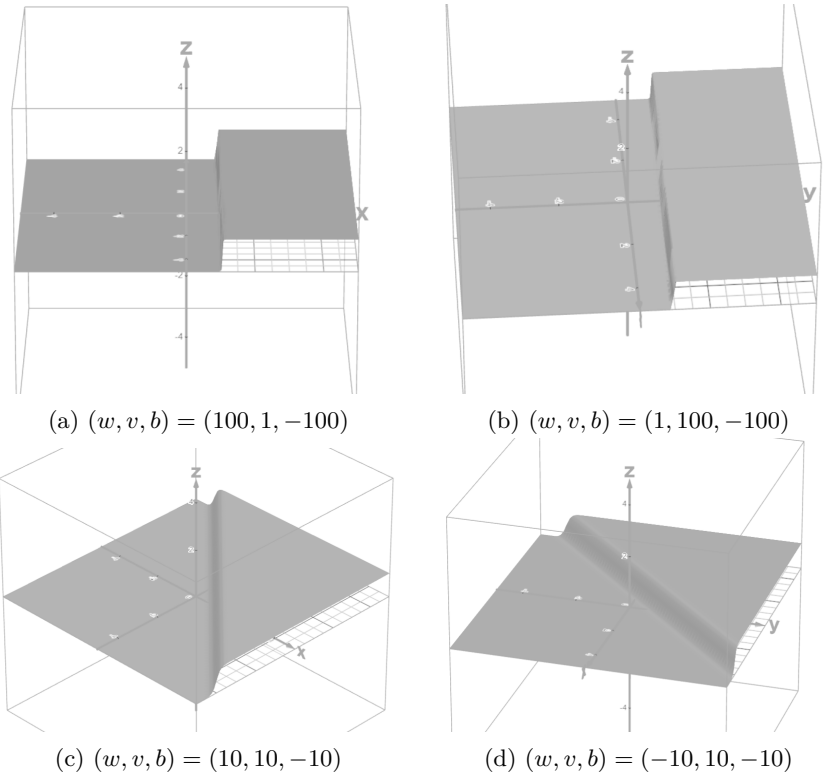
$$NN(x) = \sum_{i=1}^{n} a_i \, \sigma(w_{i1}x_1 + w_{i2}x_2 + b_i) + c,$$

with $a_i \geq 0$ and σ a sigmoid activation.

Each unit generates a "soft" step function with discontinuity along the line

$$w_{i1}x_1 + w_{i2}x_2 + b_i = 0,$$

whose normal vector is $\eta = (w_{i1}, w_{i2})$. By varying η, we control the orientation of the discontinuity. With $w_{i1}, w_{i2} \geq 0$, orientations are restricted to angles between 0 and $\pi/2$ relative to the axes.



(a) $(w, v, b) = (100, 1, -100)$ (b) $(w, v, b) = (1, 100, -100)$

(c) $(w, v, b) = (10, 10, -10)$ (d) $(w, v, b) = (-10, 10, -10)$

Figure 7: Different curves of σ function

Uniform Grid Construction. This restriction is sufficient to generate discontinuities along vertical and horizontal lines, enabling approximations on a uniform grid. For example, on the unit square $K = [0,1]^2$, let

$$0 = x_0 < x_1 < \cdots < x_N = 1, \quad 0 = y_0 < y_1 < \cdots < y_N = 1, \quad x_i = y_i = \tfrac{i}{N}.$$

By activating sigmoids along the grid lines $x = x_i$ and $y = y_j$, one obtains a step function approximation of f.

To illustrate the ideas, we propose to graphically show the case of a 3×3 subdivision.

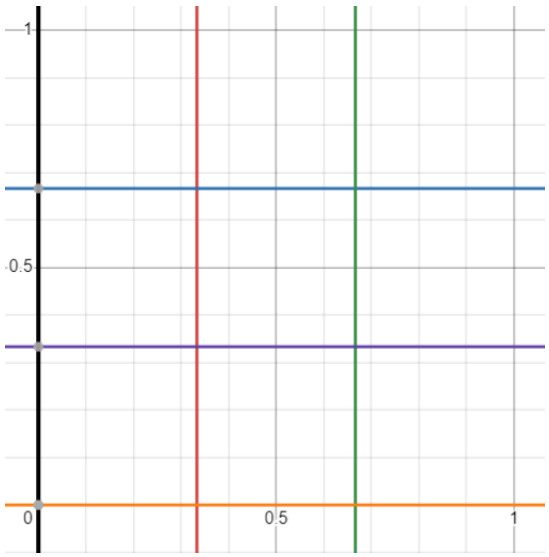


Figure 8: Subdivision of the unit square

The idea is to activate sigmoids on each vertical line $x = \frac{i}{N}$ and horizontal line $y = \frac{i}{N}$ for all $i \in \{0, 1, \ldots, N-1\}$.

This will give figures as follows:

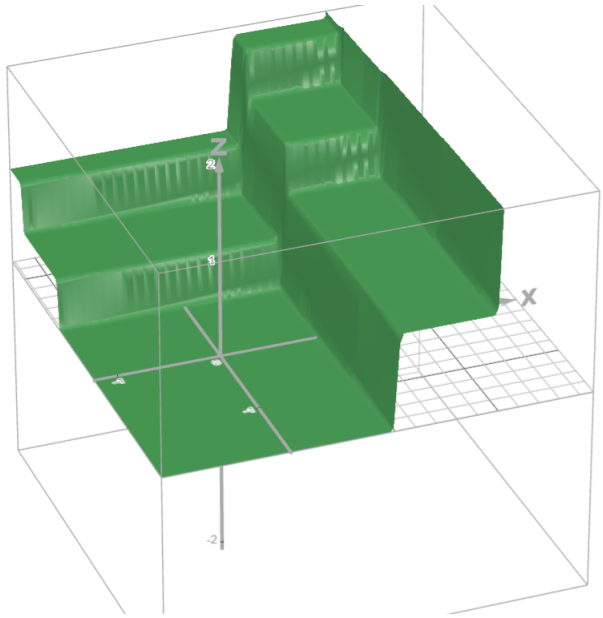


Figure 9: Example of the result of the sigmoid-based approximation on a uniform grid.

Limitation. Not all isotonic step functions can be realized this way. For example, in a 2×2 subdivision, the function represented by the matrix

$$\begin{bmatrix} 0 & 9 \\ 1 & 8 \end{bmatrix}$$

is isotonic but cannot be generated by positive sigmoids restricted to vertical and horizontal partitions.

Proposition 4.3. *The positive cone generated by axis-aligned sigmoids is not surjective onto the set of isotonic step functions.*

Recoverable Class. The functions that can be recovered are precisely the *separable isotonic functions*, i.e., those decomposable as

$$f(x, y) = g(x) + h(y),$$

with g and h increasing.

4.3 Toward a General Construction: Level Sets

Motivation. Our previous construction generated a "wave" by discriminating with the linear equation

$$w \cdot x + v \cdot y + b = 0, \quad \text{equivalently } \langle w, x \rangle + b = 0.$$

This restricted us to axis-aligned partitions, namely vertical or horizontal boundaries ($x = C$ or $y = C$). As shown, such constraints are insufficient to achieve surjectivity onto the set of isotonic functions.

To address this limitation, we propose to *deform these linear boundaries monotonically.* The general idea of activating sigmoids "by slices" is preserved; however, the slices are now curved in a monotone fashion. This ensures that the neural network activates the correct spatial regions while maintaining isotonicity.

The natural mathematical framework for this approach is given by *level sets.*

Definition 4.4 (Level Set)**.** *Let $f : \mathbb{R}^n \to \mathbb{R}$ be a real-valued function. A* level set *of f associated with a value $c \in \mathbb{R}$ is defined as*

$$L_c = \{x \in \mathbb{R}^n \mid f(x) = c\}.$$

If f is continuously differentiable, then $\nabla f(x)$ is orthogonal to the level set L_c at every regular point x.

Characterization for Isotonic Functions. For an isotonic function $f : \mathbb{R}^n \to \mathbb{R}$, level sets satisfy two important properties:

1. Distinct level sets are disjoint (true for any continuous function).
2. A level set cannot form a closed loop, since monotonicity requires that values increase consistently along isotone directions.

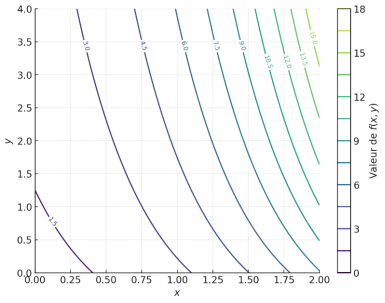


Figure 10: Level sets of the function $\exp(x) \cdot \sqrt{y + 1}$

Lemma 4.5 (level set of isotonic function)**.** *A level set of an isotonic function is necessarily decreasing.*

Proof Otherwise, there will exist an isotonic path from (x_1^1, x_2^1) to (x_1^2, x_2^2), passing through (a, b), where (x_1^1, x_2^1) and (x_1^2, x_2^2) belongs to l, but $(a, b) \notin l$. This leads to a contradiction.

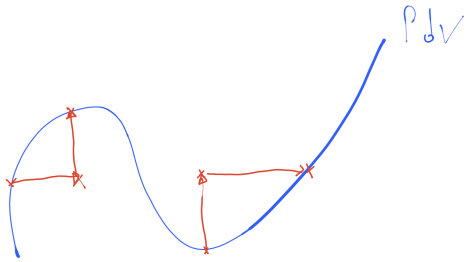


Figure 11: Illustration of the fact that a level set is decreasing.

4.4 Architecture of the Positive Neural Network

The overall construction follows the same principles as in the one-dimensional case. We begin by ordering the level sets of f in increasing order:

$$l_1 < l_2 < \cdots < l_p,$$

such that

$$\forall i \in \{1, \ldots, p-1\}, \quad \|f|_{l_i} - f|_{l_{i+1}}\| \leq \varepsilon.$$

The strategy is then to activate a sigmoid unit (with appropriately chosen amplitude) whose decision boundary coincides with each level set l_i.

Important. By extending f appropriately along each axis, we may assume without loss of generality that every level set intersects both the x- and y-axes.

Question. How can we ensure that a sigmoid has an activation boundary corresponding to a level set l ?

Classification Layer. To each level set l_i, we associate a classifier that outputs 0 below the line l_i and 1 above it. This yields an intermediate layer of classifiers, which are then aggregated by the network.

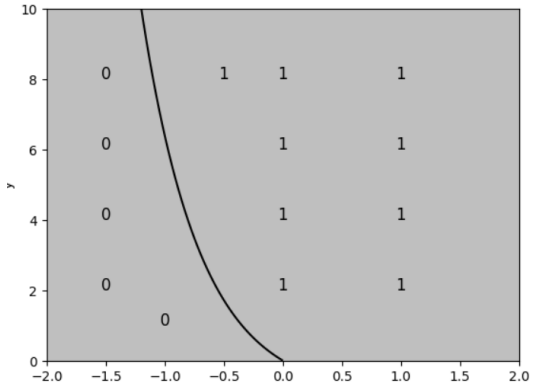


Figure 12: Example of a classifier having as a boundary a level set

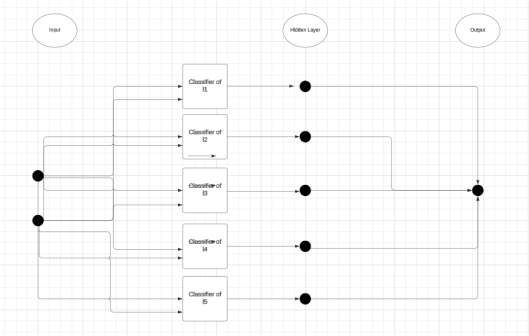


Figure 13: Proposed architecture for a neural network

4.4.1 Constructing Classifiers with Positive Weights

The key remaining question is whether classifiers with decision boundaries aligned to level sets can be implemented using strictly positive weights. We address this question in two steps: first for convex decreasing boundaries, and then for the general case.

Lemma 4.6 (Convex Decreasing Boundary)**.** *Let $l \subset \mathbb{R}^2$ be a convex decreasing curve. Then there exists a positive neural network that classifies points below l as* 0 *and points above l as* 1*.*

Proof. Let's consider a given decreasing and convex line l. Its tangents have a negative slope. The equation of a tangent line can be written as:

$$w_1 \cdot x_1 + w_2 \cdot x_2 + b = 0, \quad \text{with} \quad w_1, w_2 \geq 0.$$

Furthermore, convexity of the curve implies that all tangents are supporting lines (below l), and thus the region bounded by the level curve can be approximated by the envelope of these tangent lines.

In order to approximate the curve with an error of at most ε, we construct an interpolated function using its tangents. It's clear that each tangent divides the plane into two regions: a region above the curve, assigned the value 1, and a region below the curve, assigned the value 0.

Therefore, the true region above the level curve is approximated (within ε by the intersection of the half-spaces defined by the tangents, where the positive half-space corresponds to a value of 1. With (p) tangents, a point (x_1, x_2) is labeled 1 if and only if:

$$\text{logits} = \sum_{i=1}^{p} \big(\sigma\big(w_{1,i}x_1 + w_{2,i}x_2 + b_i\big)\big) \approx p,$$

where σ is a sigmoid activation function.

Then, we apply another sigmoid function to label the area with logits less than p to 0, and the other points to 1. As a result, we obtain a positive classifier whose decision boundary corresponds to the decreasing and convex curve l.

□

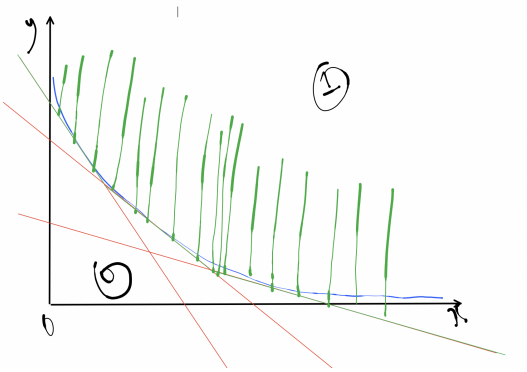


Figure 14: Case of a decreasing convex function.

Lemma 4.7. *Given a decreasing curve C that divides the space into two regions C^1 and C^0. Then C^1 could be recovered using the upper part of several decreasing convex boundaries.*

Proof :

Consider a decreasing curve C in the plane. We define C^1 and C^0 to be the regions above and below C, respectively. Consequently, C can be associated with a continuous function $g : x \mapsto g(x)$.

For every $x_0 \in I$, we associate the curve that forms a right angle :

$$C_{x_0} = \{(x_0, y) \mid y \geq g(x_0)\} \cup \{(x, g(x_0)) \mid x \geq x_0\}.$$

Then, we are going to slightly modify this curve in order to ensure its association with a decreasing convex function. A slight adjustment of its angle is sufficient.

We denote the modified version $\widetilde{C_{x_0}}$. The decreasing nature of the original curve C ensures that these two curves will never intersect!

Hence, it's easy to see that the zone C^1 above C is :

$$C^1 = \bigcup_{x_0 \in I} \widetilde{C^1_{x_0}}.$$

As a consequence of the preceding lemma, we have shown that the sets $\widetilde{C^1_{x_0}}$ can be efficiently reconstructed, given that $\widetilde{C_{x_0}}$ is a decreasing convex curve. This allows for the recovery of the set C^1.

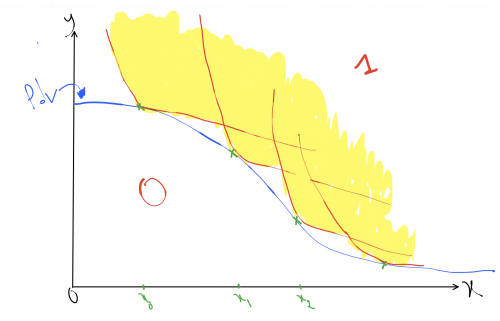


Figure 15: Cover C^1

We can prove the main result.

Theorem 4.8 (General Decreasing Boundary)**.** *Let C be a decreasing boundary. A positive classifier can be constructed to discriminate points lying below it with a value of 0 and all other points with a value of 1.*

Idea. To prove this result, we focus not on the boundary itself but on the region above it, which should be labeled 1. The key observation is that any decreasing boundary can be locally approximated from above by a collection of convex decreasing curves. By the previous lemma, each such convex boundary admits a positive classifier that correctly separates the regions above and below it.

Since the region above C is precisely the union of the regions above these convex approximations, combining the corresponding classifiers yields a network that labels all points above C as 1 and those below as 0. In this sense, we "lift" the entire region above a general decreasing boundary by successively lifting the regions above convex decreasing boundaries. □

Theorem 4.9 (Universal Approximation of Isotonic Functions)**.** *Let $f : K \subset \mathbb{R}^d \to \mathbb{R}$ be an isotonic function on a compact set K. Then for every $\varepsilon > 0$, there exists a two-layer neural network with nonnegative weights and monotone activation functions such that*

$$\sup_{x \in K} |f(x) - F(x)| < \varepsilon.$$

In other words, positive two-layer neural networks are universal approximators for isotonic functions.

Idea. The first layer identifies the level sets, while the second layer lifts these pieces to the correct heights, just as in the 1D case. □

5 Experiments

5.1 How can the positivity of the neural network be ensured?

5.1.1 Projections

We could, at each step, project each weight matrix W_i into the following set :

$$\mathcal{D}_i = \left\{ W_i \in \mathbb{R}^{N_i \times N_{i-1}} \mid W_i \geq 0 \right\}.$$

5.1.2 Map $\mathbb{R}$ into $\mathbb{R}_+$

Another approach consists of applying a **bijective function** $f : \mathbb{R} \to \mathbb{R}^+$ to the matrices, which maps the matrices to the space of matrices with strictly positive entries. For example, for a 2×2 matrix and choosing the function $f = \exp$ or even the sofIt plus operator $f = \log(1 + \exp)$, we would have:

$$\begin{bmatrix} a & b \\ c & d \end{bmatrix} \longrightarrow \begin{bmatrix} \log(1+e^a) & \log(1+e^b) \\ \log(1+e^c) & \log(1+e^d) \end{bmatrix}.$$

The parameter set is then defined as $p = \{a, b, c, d\}$. Thus, the resulting matrix (W) will have strictly positive entries for any modification of the parameters.

The use of the exponential function also mitigates the vanishing gradient problem, which is a significant advantage.

6 Extension to monotone operators

The question is whether the universal approximation of isotonic functions by positive networks extends to monotone operators. An intuitive understanding of monotone operators is a prerequisite.

References

[1] Hoefling H. Tibshirani R. Tibshirani, R. J. Nearly isotonic regression. *Technometrics*, 53(1):54–61, 2010.