Trainable and explainable simplicial map neural networks

Eduardo Paluzo-Hidalgo [a,*], Rocio Gonzalez-Diaz [b], Miguel A. Gutiérrez-Naranjo [c]

[a] *Department of Quantitative Methods, Universidad Loyola Andalucía, Campus Sevilla, Dos Hermanas, Seville, Spain*
[b] *Departamento de Matemática Aplicada I, Universidad de Sevilla, Seville, Spain*
[c] *Departamento de Ciencias de la Computación e Inteligencia Artificial, Universidad de Sevilla, Seville, Spain*

ARTICLE INFO

ABSTRACT

Simplicial map neural networks (SMNNs) are topology-based neural networks with interesting properties such as universal approximation ability and robustness to adversarial examples under appropriate conditions. However, SMNNs present some bottlenecks for their possible application in high-dimensional datasets. First, SMNNs have precomputed fixed weight and no SMNN training process has been defined so far, so they lack generalization ability. Second, SMNNs require the construction of a convex polytope surrounding the input dataset. In this paper, we overcome these issues by proposing an SMNN training procedure based on a support subset of the given dataset and replacing the construction of the convex polytope by a method based on projections to a hypersphere. In addition, the explainability capacity of SMNNs and effective implementation are also newly introduced in this paper.

1. Introduction

In recent years, Artificial Intelligence (AI) methods in general and Machine Learning methods in particular have reached success in real-life problems that were unexpected only a few years ago. Many different areas have contributed to this development. Among them, we can cite the research on new theoretical algorithms, the increasing computational power of the latest generation hardware, and the rapid access to a huge amount of data. Such a combination of factors leads to the development of increasingly complex self-regulated AI methods.

Many AI models currently used are based on backpropagation algorithms, which train and regulate themselves to achieve a goal, such as classification, recommendation, or prediction. These self-regulating models achieve some kind of knowledge as they successfully evaluate test data independent of the data used to train them. Nonetheless, such knowledge is usually expressed in a non-human-readable way.

To fill the gap between the recent development of AI models and their social use, many researchers have focused on the development of Explainable Artificial Intelligence (XAI), which consists of a set of techniques to provide clear, understandable, transparent, intelligible, trustworthy, and interpretable explanations of the decisions, predictions, and reasoning processes made by the AI models, rather than just presenting their output, especially in domains where AI decisions can have significant consequences on human life. A global taxonomy of interpretable AI with the aim of unifying terminology to achieve clarity and efficiency in the definition of

regulations for the development of ethical and reliable AI can be found in [1]. Moreover, a nice introduction and general vision can be found in [2]. Another clarifying paper with definitions, concepts, and applications of XAI in [3].

The so-called Simplicial Map Neural Networks (SMNNs) were introduced in [4] as a constructive approach to the problem of approximating a continuous function on a compact set in a triangulated space. Since the original aim of the definition of SMNNs was focused on building a *constructive* approach, the computation of their weights was not based on an optimization process, as usual in neural networks, but on a deterministic calculus. The architecture of SMNNs and the computation of the set of weights are based on a combinatorial topology tool called simplicial maps. Moreover, SMNNs can be used for classification and can be constructed to be robust to adversarial examples [5]. Besides, their architecture can be reduced while maintaining accuracy [6], being invariant to transformation if the transformation preserves the barycentric coordinates (scale, rotation, symmetries, etc.). As defined in [4,6,5], SMNNs are built as a two-hidden-layer feed-forward network where the set of weights is precomputed based on the calculation of a triangulated convex polytope surrounding the input data. As other approximations to continuous functions with arbitrary precision (see, for example, [7]), SMNNs have fixed weights, which means that the weights depend only on the triangulation made with the points of the dataset as the *support* set and no training process is applied.

Summing up, some of the limitations of the SMNNs until now are that they are costly to calculate since the number of neurons is proportional to the number of simplices of the triangulation supported on the input dataset, and they suffer from overfitting and therefore not generalize well. These aspects make SMNNs not used in practice so far, although the idea of relating simplicial maps to neural networks is disruptive and provides a new bridge that can enrich both areas.

In this paper, we propose a method to make SMNNs efficient by reducing their size (in terms of the number of neurons that depends on the vertices of the triangulation) and that successfully makes SMNNs trainable and with generalization ability. Besides, we also present a study of the selection of the vertices from which we obtain the triangulation. Although SMNNs consider the vertices of a simplex as part of the necessary information for the classification task, the approach presented in this paper is far from the classic Machine Learning instance-based methods. Such methods rely on a deterministic computation based on distances, but, in the approach presented in this paper, the computation of the weights is the result of an optimization method in a probability distribution space. Finally, from an XAI point of view, we will see in this paper that SMNNs are explainable models since all decision steps to compute the output of SMNNs are understandable and transparent, and therefore trustworthy.

The paper is organized as follows. First, some concepts of computational topology and the definition of SMNNs are recalled in Section 2. Next, in Section 3 we develop several technical details needed for the SMNN training process, which will be introduced in Section 4. Section 5 is devoted to the explainability of the model. Section 6 is devoted to discussion and limitations. Finally, the paper ends with some experiments and conclusions.

2. Background

In this section, we assume that the reader is familiar with the basic concepts of computational topology. For a comprehensive presentation, we refer to [8].

2.1. Simplicial complexes

Consider a finite set of points $V = \{v^1, \ldots, v^\beta\} \subset \mathbb{R}^n$ whose elements will be called vertices. A subset

$$\sigma = \langle v^{i_0}, v^{i_1}, \ldots, v^{i_d} \rangle$$

of V with $d + 1$ vertices (in general position) is called a d-simplex. The convex hull of the vertices of σ will be denoted by $|\sigma|$ and corresponds to the set:

$$\left\{ x \in \mathbb{R}^n : x = \sum_{j \in [\![0,d]\!]} b_j(x) v^{i_j} \right\}$$

where $[\![a,b]\!] = \{a, a+1, \ldots, b\}$ for $a < b \in \mathbb{Z}$, and

$$b(x) = (b_0(x), b_1(x), \ldots, b_d(x))$$

are called the barycentric coordinates of x with respect to σ, and satisfy that:

$$\sum_{j \in [\![0,d]\!]} b_j(x) = 1 \text{ and } b_j(x) \geq 0 \ \forall j \in [\![0,d]\!].$$

The barycentric coordinates of x can be interpreted as masses placed at the vertices of σ so x is the center of mass. All these masses are positive if and only if x is inside σ. For example, let us consider the 1-simplex $\epsilon = \langle v^{i_0}, v^{i_1} \rangle$ which is composed of two vertices of V. Then $|\epsilon|$ is the set of points in $\mathbb{R}^n$ corresponding to the edge with endpoints v^{i_0} and v^{i_1}, and if, for example, $b(x) = (0.5, 0.5)$ then x is the midpoint of $|\epsilon|$.

A simplicial complex K with vertex set V consists of a finite collection of simplices satisfying that if $\sigma \in K$ then either $\sigma = \langle v \rangle$ for some $v \in V$ or any face (that is, a nonempty subset) of σ is a simplex of K. Furthermore, if $\sigma, \mu \in K$ then $|\sigma| \cap |\mu| = \emptyset$ or $|\sigma| \cap |\mu| = |\gamma|$ for some $\gamma \in K$. The set $\bigcup_{\sigma \in K} |\sigma|$ will be denoted by $|K|$. A maximal simplex of K is a simplex that is not the face of any other
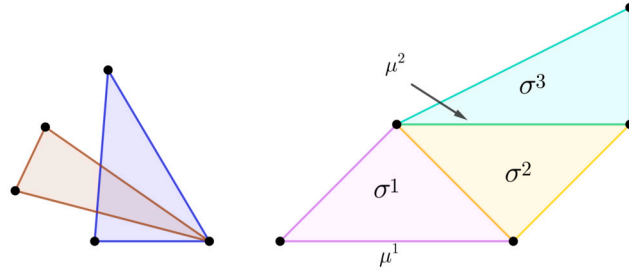
Fig. 1. On the left, two triangles that do not intersect in a common face (an edge or a vertex). On the right, the geometric representation $|K|$ of a pure 2-simplicial complex K composed of three maximal 2-simplices (the triangles σ^1, σ^2 and σ^3). The edge μ^2 is a common face of σ^2 and σ^3. The edge μ^1 is a face of σ^1.

simplex of K. If the maximal simplices of K are all d-simplices then K is called a pure d-simplicial complex. These concepts are illustrated in Fig. 1.

The barycentric coordinates of x with respect to the simplicial complex $|K|$ are defined as the barycentric coordinates of x with respect to $\sigma \in K$ such that $x \in |\sigma|$. Let us observe that the barycentric coordinates of $x \in |K|$ are unique.

An example of simplicial complexes is the Delaunay triangulation $\mathrm{Del}(V)$ defined from the Voronoi diagram of a given finite set of vertices V. The following result extracted from [9, page 48] is just an alternative definition of Delaunay triangulations.

The empty ball property [9]: Any subset σ of V is a simplex of $\mathrm{Del}(V)$ if and only if $|\sigma|$ has a circumscribing open ball empty of points of V.

2.2. Simplicial maps

Let K be a pure n-simplicial complex and L a pure k-simplicial complex with vertex sets V and W, respectively. The map $\varphi^{(0)} : V \to W$ is called a *vertex map* if it satisfies that the set obtained from $\left\{\varphi^{(0)}(v^{i_0}), \dots, \varphi^{(0)}(v^{i_d})\right\}$ after removing duplicated vertices is a simplex in L whenever $\langle v^{i_0}, \dots, v^{i_d}\rangle$ is a simplex in K. The vertex map $\varphi^{(0)}$ always induces a continuous function, called a *simplicial map* $\varphi : |K| \to |L|$, which is defined as follows. Let $b(x) = (b_0(x), \dots, b_n(x))$ be the barycentric coordinates of $x \in |K|$ with respect to $\sigma = \langle v^{i_0}, \dots, v^{i_n}\rangle \in K$. Then

$$\varphi(x) = \sum_{j \in \llbracket 0,n \rrbracket} b_j(x)\varphi^{(0)}(v^{i_j}).$$

Let us observe that $\varphi(x) = \varphi^{(0)}(x)$ if $x \in V$.

A special kind of simplicial map used to solve classification tasks will be introduced in the next subsection.

2.3. Simplicial maps for classification tasks

Next, we will show how a simplicial map can be used to solve a classification problem (see [5] for details). From now on, we will assume that the input dataset is a finite set of points V in $\mathbb{R}^n$ together with a set of k labels Λ such that each $v \in V$ is tagged with a label λ taken from Λ.

Firstly, the intuition is that the space surrounding the dataset is labelled as *unknown*. For this, we add a new label to Λ, called *unknown* label, and a one-hot encoding representation $W^{k+1} \subset \mathbb{R}^{k+1}$ of these $k + 1$ labels being:

$$W^{k+1} = \left\{\ell^j = (0, \overset{j}{\dots}, 0, 1, 0, \overset{k-j}{\dots}, 0) : j \in \llbracket 1, k \rrbracket\right\},$$

where the one-hot vector ℓ^j encodes the j-th label of Λ for $j \in \llbracket 1, k \rrbracket$ and where ℓ^0 encodes the *unknown* label.

We now consider a convex polytope $\mathcal{P}$ with a vertex set P surrounding the set V. The polytope $\mathcal{P}$ always exists since V is finite. Next, we define a map $\varphi^{(0)} : V \cup P \to W^{k+1}$ mapping each vertex $v \in V$ tagged with label λ to the one-hot vector in W^{k+1} that encodes the label λ. The vertices of P are sent to the vertex ℓ^0. Observe that $\varphi^{(0)}$ is a vertex map.

Let L denote the simplicial complex with vertex set W^{k+1} consisting of only one maximal k-simplex and let $\mathrm{Del}(V \cup P)$ denote the Delaunay triangulation computed for the set of points $V \cup P$. Note that $|\mathrm{Del}(V \cup P)| = \mathcal{P}$. The simplicial map $\varphi : \mathcal{P} \to |L|$ is induced by the vertex map $\varphi^{(0)}$ as explained in Subsection 2.2.

Remark 1. The space $|L|$ can be interpreted as the discrete probability distribution space Ω^{k+1} with $k + 1$ variables.

As an example, in Fig. 2, on the left, we can see a dataset with four points $V = \{b, c, k, d\}$, labelled red and blue. The green points $P = \{a, e, g, f\}$ are the vertices of a convex polytope $\mathcal{P}$ containing V and are sent by $\varphi^{(0)}$ to the green vertex ℓ^0 on the right. The simplicial complex $K = \mathrm{Del}(V \cup P)$ is drawn on the left and consists of ten maximal 2-simplices. On the right, the simplicial complex L consists of one maximal 2-simplex. The dotted arrows illustrate some examples of $\varphi : \mathcal{P} \to |L|$.
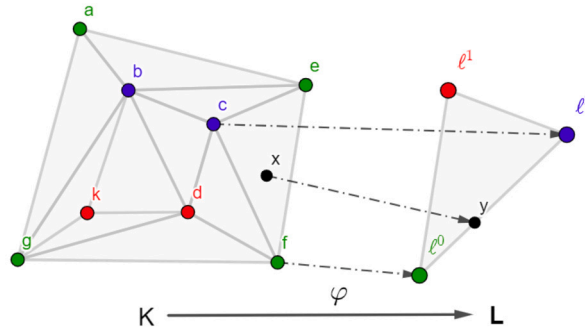
Fig. 2. Illustration of a simplicial map for a classification task.

2.4. Simplicial map neural networks

Artificial neural networks can be seen as parametrized real-valued mappings between multidimensional spaces. Such mappings are the composition of several maps (usually many of them) that can be structured in layers. In [5], the simplicial map φ defined above was represented as a two-hidden-layer feed-forward neural network $\mathcal{N}_\varphi$. This kind of artificial neural network is called *simplicial map neural network* (SMNN).

In the original definition [5], the first hidden layer of an SMNN computes the barycentric coordinates of $\mathrm{Del}(V \cup P)$. However, we will see here that if we precompute the barycentric coordinates, we can simplify the architecture of $\mathcal{N}_\varphi$ as follows.

As before, consider an input dataset consisting of a finite set $V \subset \mathbb{R}^n$ endowed with a set of k labels and a convex polytope $\mathcal{P}$ with a set of vertices P surrounding V. Let $\mathrm{Del}(V \cup P)$ be the Delaunay triangulation with vertex set

$$V \cup P = \{\omega^1, \dots, \omega^\alpha\} \subseteq \mathbb{R}^n.$$

Then, $\varphi^{(0)} : V \cup P \to W^{k+1}$ is a vertex map. Let us assume that, given $x \in \mathcal{P}$, we precompute the barycentric coordinates $b(x) = (b_0(x), \dots, b_n(x)) \in \mathbb{R}^{n+1}$ of x with respect to the n-simplex $\sigma = \langle \omega^{i_0}, \dots, \omega^{i_n} \rangle \in \mathrm{Del}(V \cup P)$ such that $x \in |\sigma|$, and that we also precompute the vector

$$\xi(x) = (\xi_1(x), \dots, \xi_\alpha(x)) \in \mathbb{R}^\alpha$$

satisfying that, for $t \in [\![1, \alpha]\!]$, $\xi_t(x) = b_j(x)$ if $i_j = t$ for some $j \in [\![0, n]\!]$. Let us remark that $\xi(x)$ should be a column vector, but we will use row notation, for simplicity.

The SMNN $\mathcal{N}_\varphi$ induced by φ that predicts the h-label of x, for $h \in [\![0, k]\!]$, has the following architecture:

- The number of neurons in the input layer is α.
- The number of neurons in the output layer is $k + 1$.
- The set of weights is represented as a $(k + 1) \times \alpha$ matrix $\mathcal{M}$ such that the j-th column of $\mathcal{M}$ is $\varphi^{(0)}(\omega^t)$ for $t \in [\![1, \alpha]\!]$.

Then,

$$\mathcal{N}_\varphi(x) = \mathcal{M} \cdot \xi(x).$$

Observe that as defined so far, the SMNN weights are precomputed. Furthermore, the computation of the barycentric coordinates of the points around V implies the calculation of the convex polytope $\mathcal{P}$ surrounding V. Finally, the computation of the Delaunay triangulation $\mathrm{Del}(V \cup P)$ is costly if $V \cup P$ has many points since its time complexity is $O(n \log n + n^{\lceil \frac{d}{2} \rceil})$ (see [9, Chapter 4]).

In the next sections, we will propose some techniques to overcome the SMNN construction drawbacks while maintaining its advantages. We will see that one way to overcome the computation of the convex polytope $\mathcal{P}$ is to consider a hypersphere S^n instead. We will also see how to avoid the use of the artificially created *unknown* label. Furthermore, to reduce the cost of Delaunay computation and add trainability to $\mathcal{N}_\varphi$ to avoid overfitting, a subset $U \subset V$ will be considered. The set V will be used to train and test a map $\varphi_U^{(0)} : U \to \mathbb{R}^k$. Such a map will induce a continuous function $\varphi_U : B^n \to |L|$ which approximates φ.

3. The *unknown* boundary and the function φ_U

In this section, we will see how to compute a function φ_U that approximates the simplicial map φ and avoids the computation of the convex polytope $\mathcal{P}$ and the artificial consideration of the *unknown* label, reducing, at the same time, the computation of the Delaunay triangulation used to construct SMNNs. The general description of the methodology is described in Algorithm 1.

First, let us compute a hypersphere surrounding V. One of the simplest ways to do that is to translate V so that its center of mass is placed at the origin $o \in \mathbb{R}^n$. Then, the hypersphere
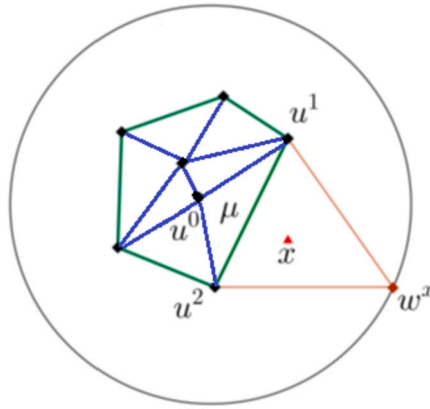
Fig. 3. An example of the point w^x computed from x and the $(n-1)$-simplex $\mu = \langle u^1, u^2 \rangle \in \Gamma$ such that $x \in |\sigma|$ for $\sigma = \langle w^x, u^1, u^2 \rangle$.

$$S^n = \{w \in \mathbb{R}^n : ||w|| = R\}$$

such that $R > \max\{||v|| : v \in V\}$ satisfies that S^n surrounds V. Second, let us assume that we have selected a subset $U = \{u^1, \ldots, u^m\} \subseteq V$ (we will compare different strategies to select U in Section 7) and that we have computed $\text{Del}(U)$. Then, we have that

$$V \subset B^n = \{x \in \mathbb{R}^n : ||x|| \leq R\} \quad \text{and} \quad o \in |\text{Del}(U)|.$$

Let us consider the boundary of $\text{Del}(U)$, denoted as $\delta \text{Del}(U)$, which consists of the set of $(n-1)$-simplices that are faces of exactly one maximal simplex of $\text{Del}(U)$.

Now, let us define $\xi_U(x) \in \mathbb{R}^m$ for any $x \in B^n$ as follows. Given $x \in B^n$, to find the n-simplex $\sigma = \langle \omega^0, \ldots, \omega^n \rangle$ such that $x \in |\sigma|$, we have to consider two cases: $x \in |\text{Del}(U)|$ and $x \notin |\text{Del}(U)|$.

If $x \in |\text{Del}(U)|$ then σ is the n-simplex in $\text{Del}(U)$ such that $x \in |\sigma|$. If $x \notin |\text{Del}(U)|$ then σ is a new n-simplex defined by the vertices of an $(n-1)$-simplex of $\delta \text{Del}(U)$ and a new vertex consisting of the projection of x to S^n. Specifically, if $x \notin |\text{Del}(U)|$ then σ is computed in the following way:

1. Consider the set

   $$\Gamma = \left\{ \mu \in \delta \text{Del}(U) : (N \cdot u^{i_0} + c)(N \cdot x + c) < 0 \right\}$$

   where N is the vector normal to the hyperplane containing $\mu = \langle u^{i_1}, \ldots, u^{i_n} \rangle$, $c = N \cdot u^{i_1}$, and $u^{i_0} \in U$ such that $\langle u^{i_0}, u^{i_1}, \ldots, u^{i_n} \rangle \in \text{Del}(U)$.

2. Compute the point $w^x = R \frac{x}{||x||} \in S^n$.

3. Find $\sigma = \langle w^x, u^{i_1}, \ldots, u^{i_n} \rangle$ such that

   $$\mu = \langle u^{i_1}, \ldots, u^{i_n} \rangle \in \Gamma \quad \text{and} \quad x \in |\sigma|.$$

   Observe that, by construction, μ always exists since $|\text{Del}(U)|$ is a convex polytope.

Now, let $b(x) = (b_0(x), \ldots, b_n(x)) \in \mathbb{R}^{n+1}$ be the barycentric coordinates of x with respect to σ. Then, $\xi_U(x) = (\xi_1(x), \ldots, \xi_m(x))$ is the point in $\mathbb{R}^m$ satisfying that, for $t \in [\![1, m]\!]$,

$$\xi_t(x) = \begin{cases} b_j(x) & \text{if } u^t = \omega^j \text{ for some } j \in [\![0, n]\!], \\ 0 & \text{otherwise.} \end{cases}$$

Observe that $\xi_U(x)$ always exists and is unique. An example of points x and w^x and simplex μ is shown in Fig. 3 and Example 1.

Let us observe that, thanks to the new definition of $\xi_U(x)$ for $x \in B^n$, if we have a map $\varphi_U^{(0)} : U \to \mathbb{R}^k$ then it induces a continuous function $\varphi_U : B^n \to |L|$ defined for any $x \in B^n$ as:

$$\begin{aligned} \varphi_U(x) &= \text{softmax}\left(\sum_{t \in [\![1, m]\!]} \xi_t(x) \varphi_U^{(0)}(u^t) \right) \\ &= \text{softmax}(\mathcal{M}_U \cdot \xi_U(x)) \end{aligned}$$

where for $z = (z_1, \ldots, z_k) \in \mathbb{R}^k$,

$$\text{softmax}(z) = \left(\frac{e^{z_1}}{\sum_{h \in [\![1, k]\!]} e^{z_h}}, \ldots, \frac{e^{z_k}}{\sum_{h \in [\![1, k]\!]} e^{z_h}} \right).$$
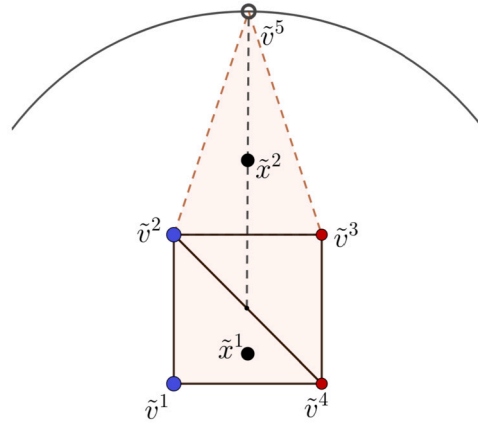
Fig. 4. The relative positions of the vertices $\tilde{v}^i$ for $i \in [\![1, 5]\!]$ and the points $\tilde{x}^1$ and $\tilde{x}^2$ of Example 1.

Let us observe that, to obtain a categorical distribution from $\varphi_U(x) \in \mathbb{R}^k$, we could just divide each of its coordinates by the total sum. However, $\mathrm{softmax}$ is introduced here to obtain a simplified formula for the gradient descent algorithm as shown in Theorem 1.

Example 1. Let us consider

$$V = \left\{ v^1 = \left(\tfrac{1}{2}, \tfrac{1}{2}\right), v^2 = \left(\tfrac{1}{2}, 1\right), v^3 = \left(1, \tfrac{1}{2}\right), v^4 = (1, 1) \right\}$$

with label $\lambda^1 = 0$ for v^i with $i = 1, 2$ and $\lambda^2 = 1$ for v^i with $i = 3, 4$. Firstly, we translate V so that the center of mass of V is the origin $o \in \mathbb{R}^2$. The translated dataset $\tilde{V}$ is

$$\left\{ \tilde{v}^1 = \left(\tfrac{-1}{4}, \tfrac{-1}{4}\right), \tilde{v}^2 = \left(\tfrac{-1}{4}, \tfrac{1}{4}\right), \tilde{v}^3 = \left(\tfrac{1}{4}, \tfrac{-1}{4}\right), \tilde{v}^4 = \left(\tfrac{1}{4}, \tfrac{1}{4}\right) \right\}.$$

Let us consider $x^1 = \left(\tfrac{3}{4}, \tfrac{3}{5}\right)$ and $x^2 = \left(\tfrac{3}{4}, \tfrac{5}{4}\right)$. Hence, the translated input data is $\tilde{x}^1 = \left(0, \tfrac{-3}{20}\right)$ and $\tilde{x}^2 = \left(0, \tfrac{1}{2}\right)$.

To simplify the explanation of the method, consider $U = \tilde{V}$, that is, $u^i = \tilde{v}^i$ for $i \in [\![1, 4]\!]$. Then, the maximal simplices of $\mathrm{Del}(U)$ are $\tilde{\sigma}^1 = \langle \tilde{v}^1, \tilde{v}^2, \tilde{v}^3 \rangle$ and $\tilde{\sigma}^1 = \langle \tilde{v}^2, \tilde{v}^3, \tilde{v}^4 \rangle$

- The matrix $\mathcal{M}_U$ is: $\begin{pmatrix} 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 \end{pmatrix}$
- Since the barycentric coordinates of $\tilde{x}^1$ with respect to $|\tilde{\sigma}^1|$ are $(0.5, 0.3, 0.2)$ then $\tilde{x}^1$ is in $|\tilde{\sigma}^1| \subset |\mathrm{Del}(U)|$ and $\xi_U(\tilde{x}^1) = (0.3, 0.2, 0, 0.5)$. Then

$$\varphi_U(\tilde{x}^1) = \mathrm{softmax}(\mathcal{M}_U \cdot \xi_U(\tilde{x}^1)) = (0.5, 0.5).$$

- On the other hand, the point $\tilde{x}^2$ is outside $|\mathrm{Del}(U)|$. Assuming that, for example, we have fixed $R = 1$, we add a new simplex $\sigma^3 = \{w^x, \tilde{v}^1, \tilde{v}^2\}$ where $w^x = \tilde{v}^5 = (0, 1)$ which is the projection of $\tilde{x}^2$ to the hypersphere of radius R centered in the origin. See Fig. 4. Then, the barycentric coordinates of $\tilde{x}^2$ with respect to σ^3 are $(0.33, 0.33, 0.33)$ and then $\xi_U(\tilde{x}^2) = (0, 0.33, 0.33, 0)$, concluding that

$$\varphi_U(\tilde{x}^2) = \mathrm{softmax}(\mathcal{M}_U \cdot \xi_U(\tilde{x}^2)) = (0.5, 0.5).$$

The pseudocode for computing $\varphi_U(x)$ is provided in Algorithm 1.
The following property holds.

Lemma 1 *(Continuity). Let $x \in B^n$. Then,*

$$\lim_{y \to x} \xi_U(y) = \xi_U(x).$$

Proof. If $x \in |\mathrm{Del}(U)|$ then the result holds due to the continuity of the barycentric coordinates transformation. If $x \notin |\mathrm{Del}(U)|$, since the origin $o \in |\mathrm{Del}(U)|$, then $||x|| \neq 0$. Therefore, for y close to x, $||y|| \neq 0$ and $w^y = R\frac{y}{||y||} \in \mathbb{R}^n$. Besides,

$$\lim_{y \to x} w^y = w^x$$

and therefore

Algorithm 1 Pseudocode to compute $\varphi_U(x)$ for $x \in B^n$ and a subset U of an input dataset V surrounded by a hypersphere of radius R.

Input: $U \subset V \subset \mathbb{R}^n$ labelled using a set of labels $\Lambda = \{\lambda^1, \ldots, \lambda^k\}$, a radius R, and a point $x \in B^n$.
Output: The value of $\varphi_U(x)$
compute Del(U)
$W^k := \left\{ \ell^j := (0, \overset{j-1}{\ldots}, 0, 1, 0, \overset{k-j}{\ldots}, 0) \text{ for } j \in [\![1, k]\!] \right\}$
init an empty matrix $\mathcal{M}_U$
init an empty vector $xi(x)$
for $u \in U$ **do**
    if λ^j is the label of u **then**
        add ℓ^j as a column of $\mathcal{M}_U$
    end if
end for
for σ maximal simplex of Del(U) **do**
    compute $b(x)$
    if $b_j \geq 0$ for all $j \in [\![0, n]\!]$ **then**
        stop: compute $\xi_U(x)$
    end if
end for
if $\xi_U(x)$ is empty **then**
    compute Γ, w^x, μ and σ
    compute $b(x)$ with respect to σ and $\xi_U(x)$
end if
$\varphi_U(x) := \mathrm{softmax}(\mathcal{M}_U \cdot \xi_U(x))$

$$\lim_{y \to x} \xi_U(y) = \xi_U(x),$$

concluding the proof. $\quad\square$

Lemma 2 (Consistence). *Let $\varphi^{(0)}$ be the map defined in Subsection 2.3. If $U = V$ and $\varphi_U^{(0)}(u) = \varphi^{(0)}(u)$ for all $u \in U$ then*

$$\arg\max \varphi_U(x) = \arg\max \varphi(x); \text{ for all } x \in \mathrm{Del}(V).$$

Proof. Let us observe that if $U = V$ and $\varphi_U^{(0)}(u) = \varphi^{(0)}(u)$ for all $u \in U$ then, for any $x \in \mathrm{Del}(V)$, we have that:

$$\varphi(x) = \sum_{t \in [\![1,m]\!]} \xi_t(x)\varphi^{(0)}(u^t) = \sum_{t \in [\![1,m]\!]} \xi_t(x)\varphi_U^{(0)}(u^t).$$

Then, $\varphi_U(x) = \mathrm{softmax}\left(\varphi(x)\right)$ and $\arg\max \varphi_U(x) = \arg\max \varphi(x)$. $\quad\square$

One of the keys to our study is the identification of the points of $\mathbb{R}^n$ allocated inside a given simplex, with the set of all probability distributions with $n + 1$ support values. In this way, the barycentric coordinates of a point can be seen as a probability distribution. From this point of view, given $x \in B^n$, then $\varphi(x)$ and $\varphi_U(x)$ are both in the set $|L|$ of probability distributions with k support points. This is why the categorical cross-entropy loss function $\mathcal{L}$ will be used to compare the similarity between φ and φ_U. Specifically, for $v \in V$, $\mathcal{L}$ is defined as:

$$\mathcal{L}(\varphi_U, \varphi, v) = - \sum_{h \in [\![1,k]\!]} y_h \log(s_h),$$

where $\varphi^{(0)}(v) = (y_1, \ldots, y_k)$ and $\varphi_U(v) = (s_1, \ldots, s_k)$.

The following lemma establishes a specific set $U \subset V$ and a function $\hat{\varphi}_U$ such that $\mathcal{L}(\varphi_U, \varphi, v) = 0$ for all $v \in V$.

Lemma 3 ($\mathcal{L}$-optimum simplicial map). *Let U be a subset of V satisfying, for all $u \in U$, that:*

1. $\varphi_U^{(0)}(u) = \varphi^{(0)}(u)$, and
2. *if $v \in V$ such that $\varphi^{(0)}(v) \neq \varphi^{(0)}(u)$ and $\langle v, u \rangle \in \mathrm{Del}(V)$ then $v \in U$.*

Then,

$$\arg\max \varphi_U(x) = \arg\max \varphi(x); \text{ for all } x \in \mathrm{Del}(U).$$

Proof. As proved in [6], under the assumptions stated in this lemma, we have that, for all $x \in <|\mathrm{Del}(U)|$:

$$\varphi(x) = \sum_{t \in [\![1,m]\!]} \xi_t(x)\varphi_U^{(0)}(u^t)$$

and then $\arg\max \varphi_U(x) = \arg\max \varphi(x)$. $\quad\square$

Unfortunately, to compute the subset U satisfying the conditions stated in Lemma 3, we need to compute the entire triangulation $\text{Del}(V)$ which is computationally expensive, as we have already mentioned above.

4. Training SMNNs

The novel idea of this paper is to learn the function $\varphi_U^{(0)}$ for any given $U \subset V$, using the gradient descent algorithm, in order to minimize the loss function $\mathcal{L}(\varphi_U, \varphi, v)$ for any $v \in V$. The following result provides an expression of the gradient of $\mathcal{L}$ in terms of the functions φ_U and φ, and the set V.

Theorem 1. *Let $U = \{u^1, \ldots, u^m\}$ be a subset with m elements taken from a finite set of points $V \in \mathbb{R}^n$ tagged with labels taken from a set of k labels. Let $\varphi_U : B^n \to |L|$ and $\varphi^{(0)} : V \to W^k$. Let us consider that*

$$\left\{ \varphi_U^{(0)}(u^t) = (p_1^t, \ldots, p_k^t) : t \in [\![1, m]\!] \right\}$$

is a set of variables. Then, for $v \in V$,

$$\frac{\partial \mathcal{L}(\varphi_U, \varphi, v)}{\partial p_j^t} = (s_j - y_j)\xi_t(v)$$

where $j \in [\![1, k]\!]$, $t \in [\![1, m]\!]$, $\varphi^{(0)}(v) = (y_1, \ldots, y_k)$ and $\varphi_U(v) = (s_1, \ldots, s_k)$.

Proof. We have:

$$\frac{\partial \mathcal{L}(\varphi_U, \varphi, v)}{\partial p_j^t} = -\frac{\partial \left(\sum_{h \in [\![1,k]\!]} y_h \log(s_h) \right)}{\partial p_j^t}$$

$$= -\sum_{h \in [\![1,k]\!]} y_h \frac{\partial \log(s_h)}{\partial p_j^t}$$

$$= -\sum_{h \in [\![1,k]\!]} y_h \frac{\partial \log(s_h)}{\partial z_j} \frac{\partial z_j}{\partial p_j^t}.$$

Since $s_h = \frac{e^{z_h}}{\sum_{t \in [\![1,k]\!]} e^{z_t}}$ then

$$\frac{\partial \log(s_h)}{\partial z_j} = \frac{\partial \log \left(\frac{e^{z_h}}{\sum_{t \in [\![1,k]\!]} e^{z_t}} \right)}{\partial z_j}$$

$$= \frac{\partial \log(e^{z_h})}{\partial z_j} - \frac{\partial \log \left(\sum_{t \in [\![1,k]\!]} e^{z_t} \right)}{\partial z_j}$$

$$= \frac{\partial z_h}{\partial z_j} - \frac{1}{\sum_{t \in [\![1,k]\!]} e^{z_t}} \sum_{t \in [\![1,k]\!]} \frac{\partial e^{z_t}}{\partial z_j}$$

$$= \delta_{hj} - \frac{e^{z_j}}{\sum_{t \in [\![1,k]\!]} e^{z_t}} = \delta_{hj} - s_j.$$

Besides, since $z_j = \sum_{h \in [\![1,m]\!]} \xi_h(v) p_j^h$ then

$$\frac{\partial z_r}{\partial p_j^t} = \sum_{h \in [\![1,m]\!]} \xi_h(v) \frac{\partial p_r^h}{\partial p_j^t} = \xi_t(v).$$

Finally,

$$\frac{\partial \mathcal{L}(\psi, \varphi, v)}{\partial p_j^t} = -\sum_{h \in [\![1,k]\!]} y_h (\delta_{hj} - s_j)\xi_t(v)$$

$$= -\xi_t(v) \left(\sum_{h \in [\![1,k]\!]} y_h \delta_{hj} - s_j \sum_{h \in [\![1,k]\!]} y_h \right)$$

$$= (s_j - y_j)\xi_t(v). \quad \square$$

Let us now see how we add trainability to the SMNN $\mathcal{N}_{\varphi_U}$ induced by φ_U. Let V be the training set and let U be a support set lying in the same space as V. First, assuming that $U = \{u^1, \ldots, u^m\}$ has m elements, then $\mathcal{N}_{\varphi_U}$ is a multiclass perceptron with an input layer with m neurons that predicts the h-th label for $h \in [\![1, k]\!]$ using the formula:

$$\mathcal{N}_{\varphi_U}(x) = \text{softmax} \left(\tilde{\mathcal{M}} \cdot \xi_U(x) \right)$$
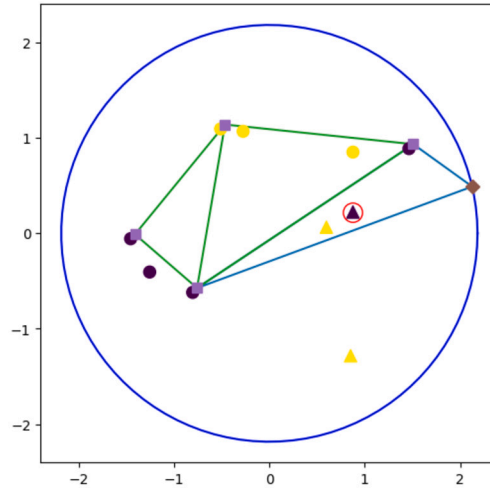
Fig. 5. Two-dimensional synthetic dataset with points divided into two classes: blue and yellow. Triangle-shaped points belong to the test set and square-shaped points belong to the support set U. The diamond-shaped point is the vertex on the hypersphere (the blue circumference) used to classify the triangle-shaped point v (surrounded by a small red circumference) outside the triangulation.

where $\tilde{\mathcal{M}} = (p_j^t)_{j \in [\![1,k]\!], t \in [\![1,m]\!]}$ is a matrix of weights and $\xi_U(x) \in \mathbb{R}^m$ is obtained from the barycentric coordinates of $x \in B^n$ as in Section 3. Let us observe that

$$\text{softmax}\left(\tilde{\mathcal{M}} \cdot \xi_U(x)\right) \in |L|.$$

The idea is to modify the initial values of

$$\varphi_U^{(0)}(u^t) = (p_1^t, \dots, p_k^t) \text{ for } u^t \in U \text{ and } t \in [\![1,m]\!],$$

in order to obtain new values for $\mathcal{N}_{\varphi_U}(v)$ for $v \in V$ in a way that the error $\mathcal{L}(\mathcal{N}_{\varphi_U}, \varphi, v)$ decreases. We will do it by avoiding recomputing $\text{Del}(U)$ or the barycentric coordinates $(b_0(v), \dots, b_n(v))$ for each $v \in V$ during the training process.

In this way, given $v \in V$, if $v \in |\text{Del}(U)|$, we compute the maximal simplex $\sigma = \langle u^{i_0}, \dots, u^{i_n} \rangle \in \text{Del}(U)$ such that $v \in |\sigma|$ and $i_h \in [\![1,m]\!]$ for $h \in [\![0,n]\!]$. If $v \notin |\text{Del}(U)|$, we compute $w \in S^n$ and the simplex $\sigma = \langle w, u^{i_1}, \dots, u^{i_n} \rangle$ such that $v \in |\sigma|$ and $i_h \in [\![1,m]\!]$ for $h \in [\![1,n]\!]$. Then we compute the barycentric coordinates $b(v)$ of v with respect to σ and the point $\xi_U(x) = (\xi_1(x), \dots, \xi_m(x)) \in \mathbb{R}^m$ as in Section 3.

Using the gradient descent algorithm, we update the variables p_j^t for $j \in [\![1,k]\!]$ and $t \in [\![1,m]\!]$ as follows:

$$p_j^t := p_j^t - \eta \frac{\partial \mathcal{L}(\mathcal{N}_{\varphi_U}, \varphi, v)}{\partial p_j^t} = p_j^t - \eta(s_j - y_j)\xi_t(v).$$

An illustrative picture of the role of each point in a simple two-dimensional binary classification problem is provided in Fig. 5. The pseudocode of the method to train SMNNs using Stochastic Gradient Descent is provided in Algorithm 2.

Algorithm 2 Pseudocode of the proposed method to train SMNNs using SGD.

Input: Dataset $V \subset \mathbb{R}^n$ surrounded by a hypersphere of radius R and a model $\mathcal{N}_{\varphi_U}$.
Parameter: $\eta > 0$
Output: The trained model $\mathcal{N}_{\varphi_U}$
init $\tilde{\mathcal{M}}$, the matrix of weights of $\mathcal{N}_{\varphi_U}$
for $v \in V$ **do**
    compute $\xi_U(v)$ as in Section 3
    for each column p of $\mathcal{M}$ **do**
        $p := p - \eta \frac{\partial \mathcal{L}(\mathcal{N}_{\varphi_U}, \varphi, v)}{\partial p}$
    end for
end for
$\mathcal{N}_{\varphi_U}(x) := \text{softmax}(\tilde{\mathcal{M}} \cdot \xi_U(x))$

---

5. Explainability

In this section, we provide insight into the explainability capability of SMNNs. In the literature, many different approaches can be found to *what is an explanation* of the prediction of an AI model. In our case, explainability will be provided based on similarities and dissimilarities of the point x to be explained with the points corresponding to the vertices of the simplex σ containing it. Based on
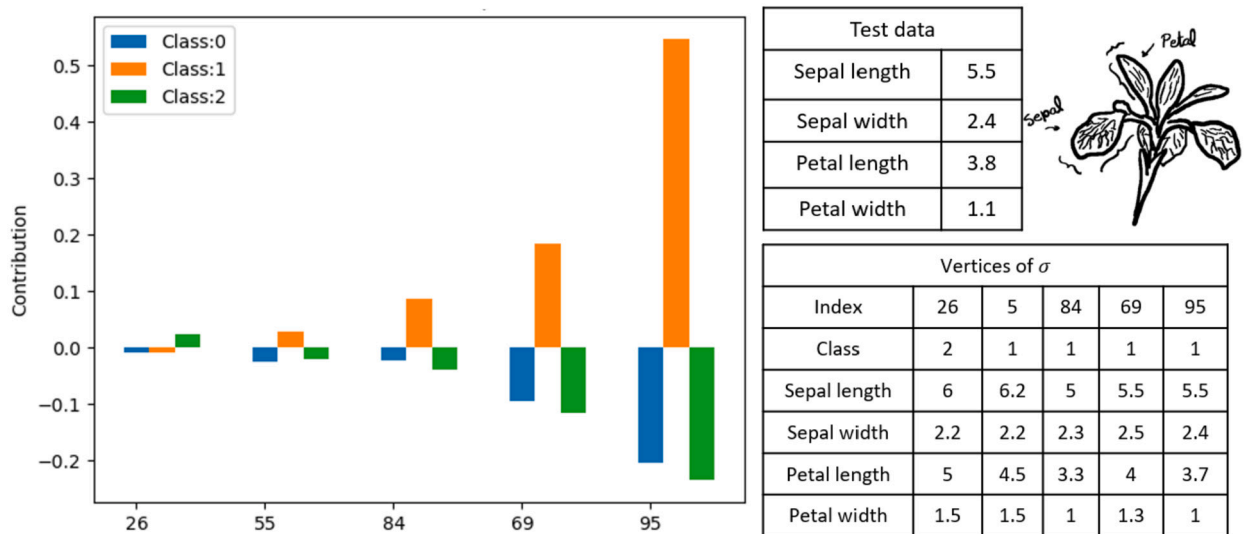
Fig. 6. Table with the five flowers taken from the training set that influence the classification of a given flower in the test set, i.e., the vertices of σ. The SMNN was trained for 1000 epochs and $R \approx 6.16$.

this idea, the barycentric coordinates of x with respect to σ can be considered indicators of how much a vertex of σ will contribute to the prediction of x. Specifically, the barycentric coordinates of x multiplied by the evaluation of the trained map $\varphi_U^{(0)}$ at the vertices of σ encode the contribution of each vertex of σ to the label assigned to x by the SMNN.

As an illustration, consider the Iris dataset[1] as a toy example and split it into a training set (75%) and a test set (25%). Since we focus on this section on explainability, let us take $U = V$, containing 112 points. Then, initialize p_j^t with a random value in $[0, 1]$, for $j \in [\![1, 4]\!]$ and $t \in [\![1, 112]\!]$.

After the training process, the SMNN reached 92% accuracy and 0.5 loss value on the test set. Once the SMNN is trained, we may be interested in determining why a certain output is given for a specific point x in the test set.

As mentioned above, the explanation for why the SMNN assigns a label to x is based on the labels of the vertices of the simplex of Del(U) containing x. Therefore, the first step is to find the maximal simplex σ that contains the point x to be explained. As an example, in Fig. 6, the point $x = (5.5, 2.4, 3.8, 1.1) \in \mathbb{R}^4$ in the test set is chosen to be explained, to which the SMNN predicts to assign class 2. The coordinates of the five vertices (u^{26}, u^{55}, u^{69}, u^{84} and u^{95}) of the simplex σ containing x together with the classes they have been assigned are shown in the table at the bottom of Fig. 6. The contribution of the class assigned to each vertex of σ to the class assigned to x by the SMNN is displayed in the bar chart, and is measured in terms of $p_j^t \cdot \xi_t(x)$ for $j \in [\![0, 2]\!]$ and $t \in \{26, 55, 69, 84, 95\}$. Let us notice that the contributions can be positive or negative. For example, the vertex with index 95 with the greatest influence on the prediction negatively affected the classification of x corresponding to the first and third class, but positively to the second class, which is the correct classification. Let us note that the Euclidean distance between points is not the only factor making a vertex of σ contribution greater. That is, even if two vertices are equally close to the point to be explained, they do not contribute the same. For example, the vertices 84 and 95 are similarly close to the test point, but their contribution is very different in magnitude.

6. Discussion and limitations

Let us remark that SMNNs are in between an instance-based method and a multilayer perceptron. The previous definition of SMNNs ([4,5]) shared advantages with instance-based methods such as the k-Nearest Neighbor Algorithm. Some of the advantages are: there was no training phase, it handles easily complex decision boundaries by barycentric subdivisions, it is effective in low-dimensional spaces, it adapts well to local patterns in the data, and the decision-making is transparent and interpretable. However, it was computationally expensive for large high-dimensional datasets due to the Delaunay triangulation computation and the required convex polytope, it suffered from overfitting and lacked generalization ability. The proposed update in this paper provides a substitute for the convex polytope which reduces the number of points needed to compute the Delaunay triangulation. Delaunay computation is also less expensive thanks to the use of a support set U.

Nevertheless, one of the main limitations of SMNNs is the need for an input triangulable space. Hence, structured data such as images need to be embedded by, for example, applying a dimensionality reduction technique such as UMAP [10] so that the dataset is a point cloud.

---

[1] https://archive.ics.uci.edu/ml/datasets/iris.

Table 1

Accuracy score and loss values obtained after training both the SMNN and a feed-forward neural network (FFNN). The experiments were repeated 100 times and the results provided are the mean of the accuracy values of the repetitions. The size m of the subset considered to compute the Delaunay triangulation also varies in each experiment depending on a parameter κ. The FFNN is composed of two hidden layers of size 32 and 16, respectively, with ReLu activation functions and an output layer with a softmax activation function. The datasets used are synthetic binary-class datasets with $n = 2, 3, 4, 5$ features.

		SMNN			FFNN	
n	κ	m	Acc.	Loss	Acc.	Loss
2	1000	3560	0.87	0.64	0.91	0.23
	100	1282	0.90	0.51		
	50	626	0.9	0.42		
	10	53	0.87	0.33		
3	1000	3750	0.76	0.66	0.8	0.61
	100	3664	0.76	0.66		
	50	3252	0.77	0.65		
	10	413	0.81	0.5		
4	50	3728	0.69	0.67	0.72	0.69
	10	1410	0.73	0.64		
	5	316	0.73	0.57		
	2	26	0.72	0.56		
5	50	3743	0.77	0.66	0.8	0.91
	10	1699	0.81	0.63		
	5	323	0.8	0.52		
	2	17	0.74	0.53		

Regarding the explainability of the model, it is not a feature-based explainability such as [11,12] and, hence, it does not provide direct insight on the importance of the features of the input data. However, it provides instance-based explainability [13]. When predicting input data, the vertices of the simplex where it belongs and their contribution to the classification give a similarity measure inferred by the SMNN which is understandable by an expert.

7. Experiments

In this section, we provide experiments that show the performance of SMNNs. In all the experiments, we split the given dataset into a training set and a test set composed of 75% and 25% of the dataset, respectively. The datasets used for experimentation are (1) a two-dimensional binary classification spiral synthetic dataset, and (2) dimension-varying binary classification synthetic datasets composed of different noisy clusters for each class (we refer to [14] for a specific description of how data is generated). All experiments were developed using a 3.70 GHz AMD Ryzen 9 5900X 12-Core Processor.

In the first two experiments, ε-representative subsets of the training set are used as the support set U for different values of ε. The notion of ε-representative sets was introduced in [15]. Specifically, a support set U is ε-representative of a set V if, for any $v \in V$, there exists $u \in U$ such that the distance between u and v is less than ε.

Let us now describe the methodology of each experiment.

First experiment: we consider a two-dimensional spiral dataset for binary classification composed of 400 two-dimensional points. We selected three different values of ε obtaining three ε-representative sets (the support sets) of size 5, 9 and 95, respectively. In Fig. 7, the spiral dataset and the three different support sets with the associated Delaunay triangulation are shown. In this case, we observed that the accuracy of the SMNNs increases with the size of the support set. We can also appreciate that the topology of the dataset is characterized by the support set, which we guess is responsible for the successful classification.

Second experiment: we consider four synthetic datasets of size 5000 using the adaptation of [14] in the scikit-learn [16] implementation. The four datasets generated have, respectively, $n = 2, 3, 4$ and 5 features. Then, the corresponding training set V obtained from each dataset and a fully connected two-hidden-layer feed-forward (32×16) neural network (FNNN) with ReLU activation functions were considered.

To train the SMNN, we use four different ε-representative sets of each V. In our experiments, the different values of ε are calculated as the maximum distance from the origin to the farthest point in the dataset plus $\frac{1}{2}$ and divided by a parameter κ. The different values for κ considered are $1000, 100, 50, 10$. Using the different values of κ we then obtain support sets U of different sizes m. For example, for $n = 2$ and $k = 1000$, we obtain a support set U of size $m = 3560$. The sizes m of the support sets U generated for $n = 2, 3, 4, 5$ and $\kappa = 1000, 100, 50, 10$ are provided in Table 1.
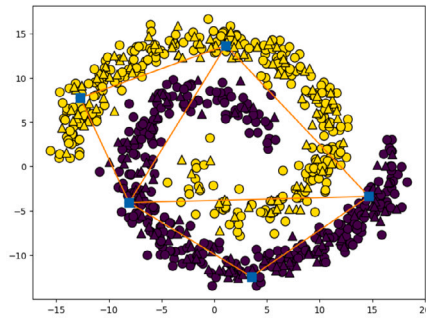
(a) Using a support set of 5 points, the SMNN reaches 80% test accuracy.
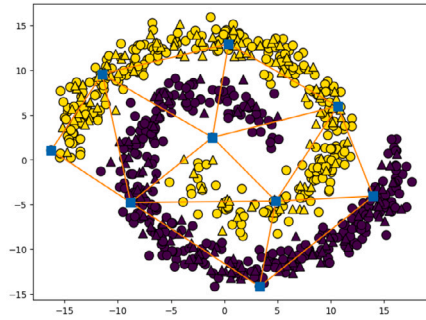
(b) Using a support set of 9 points, the SMNN reaches 93% test accuracy.
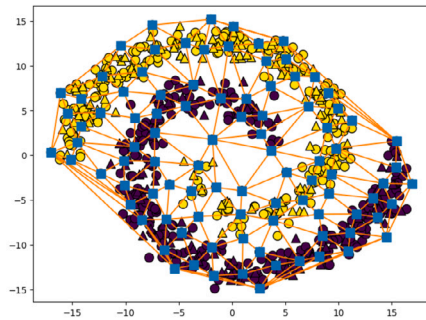
(c) Using a support set of 95 points, the SMNN reaches 99% test accuracy.

Fig. 7. Spiral dataset for binary classification. On each figure, the points in the support set are square-shaped, the points in the test set are triangle-shaped and the points in the training set are circle-shaped.

First, the SMNN was trained using the gradient descent algorithm and the cross-entropy loss function during 500 epochs for $n = 2, 3, 4, 5$ and $\kappa = 1000, 100, 50, 10$. Besides, the two-hidden-layer feed-forward neural network FFNN was trained using the Adam training algorithm [17] for $n = 2, 3, 4, 5$. Both training processes were carried out for 500 epochs. The accuracy and loss results provided in Table 1 are the mean of 100 repetitions. We can observe that both the SMNN and the FFNN have similar performance, but the SMNN generally reaches lower loss values. The variance in the results was on the order of 10^{-8} to 10^{-5} in the case of the SMNN and of 10^{-5} to 10^{-2} in the case of the FFNN.

Third experiment: we compare the performance of the SMNN depending on the choice of the support set U. To do this, we applied two different methods to choose U. On the one hand, we use the ε-representative sets previously computed in the first experiment for different values of κ. On the other hand, we use the two outlier-robust subsampling methods presented in [18] for different values of Topological Density (TD). In Fig. 8, examples of different support sets computed using the three different approaches are shown. Let us remark that the outlier-robust subsampling method can be tuned to keep outliers, which the authors call *vital landmark* set, or not, obtaining a *representative landmark* set. The two methods were tested on synthetic datasets composed of 1000 points. The SMNN was trained for 1000 epochs using the 75% of the dataset and tested in the remaining 25%. The number of features n of each dataset considered, the size m of each support set computed, and the mean accuracy and loss results of 5 repetitions
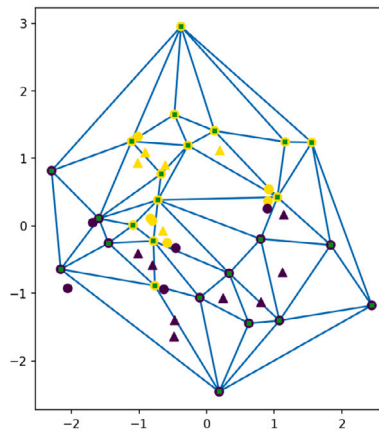
(a) ε-Representative set.
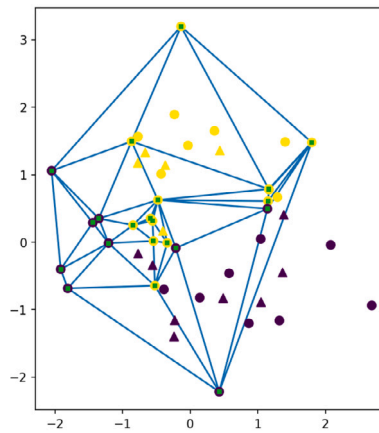
(b) Outlier-robust representative landmark set.
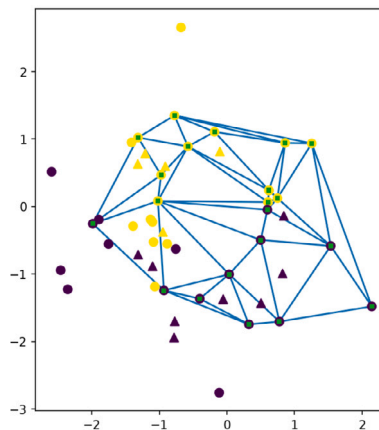
(c) Outlier-robust vital landmark set.

Fig. 8. Examples of support sets obtained using the different methods proposed in the third experiment.

when training the SMNN are provided in Table 2. In Table 3, the time for the computation of the barycentric coordinates is provided. As we can see, the time of execution does not have to be directly related to the size of the support set and it may increase if many of the points to be evaluated are outside the Delaunay triangulation of U. In this experiment, the results suggest that ε-representative datasets provide better results than the other support sets. However, thorough theoretical studies should be developed to confirm the last statement. Such a study is outside of the scope of this paper.

Table 2

Accuracy score and loss values obtained after training the SMNN. The size m of the subset considered to compute the Delaunay triangulation varies in each experiment depending on a parameter κ (for ε-representative sets) or a parameter TD (for landmark sets). The datasets used are synthetic binary-class datasets with $n = 2, 3, 4$ features.

Sampling method		Parameter	$n = 2$			$n = 3$			$n = 4$		
			m	Acc.	Loss	m	Acc.	Loss	m	Acc.	Loss
Representative sets		$\kappa = 10$	48	0.94	0.22	109	0.94	0.29	407	0.9	0.53
		$\kappa = 50$	371	0.94	0.47	730	0.95	0.53	748	0.87	0.6
		$\kappa = 100$	570	0.93	0.54	747	0.96	0.56	750	0.87	0.6
ORS	Representative landmark sets	TD = 0.1	75	0.79	0.49	75	0.9	0.35	75	0.85	0.46
		TD = 0.4	300	0.85	0.57	300	0.86	0.5	300	0.87	0.62
		TD = 0.6	450	0.86	0.52	450	0.89	0.52	450	0.86	0.55
		TD = 0.8	600	0.89	0.56	600	0.92	0.52	600	0.86	0.58
	Vital landmark sets	TD = 0.1	75	0.93	0.27	75	0.84	0.4	75	0.88	0.37
		TD = 0.4	300	0.93	0.37	300	0.88	0.46	300	0.9	0.45
		TD = 0.6	450	0.91	0.44	450	0.96	0.44	450	0.89	0.53
		TD = 0.8	600	0.93	0.5	600	0.95	0.52	600	0.87	0.57

Table 3

Time in seconds for the $\xi_U(x)$ computation for the experiments in Table 2. The values are the mean of 5 iterations. Let us remark that higher values can be expected when increasing the number of points outside the Delaunay triangulation of the support set.

Sampling method		Parameter	$n = 2$	$n = 3$	$n = 4$
Representative sets		$\kappa = 10$	0.08	0.14	0.54
		$\kappa = 50$	0.12	0.22	0.79
		$\kappa = 100$	0.14	0.24	0.79
ORS	Representative landmark sets	TD = 0.1	1.04	2.89	21.75
		TD = 0.4	0.33	0.29	7
		TD = 0.6	0.12	0.37	3.03
		TD = 0.8	0.1	0.23	1.29
	Vital landmark sets	TD = 0.1	0.17	3.22	3.87
		TD = 0.4	0.15	2.31	1.69
		TD = 0.6	0.08	0.53	1.17
		TD = 0.8	0.15	0.27	0.82

8. Conclusions

The balance between efficiency and explainability will be one of the major problems of AI in the next years. Although AI models based on network architectures and backpropagation algorithms are currently among the most successful models, they are far from providing a human-readable explanation of their outputs. On the other hand, simpler models not based on gradient descent methods usually do not provide a comparable level of performance. In this way, a trainable version of SMNNs provides a new step in filling the gap between both approaches.

Simplicial map neural networks provide a combinatorial approach to artificial intelligence. Its simplicial-based definition provides nice properties, such as easy construction and robustness capability against adversarial examples.

In this work, we have extended its definition to provide a trainable version of this architecture. The training process is based on a local search and links this model based on simplices with the most efficient methods in AI. Moreover, we have demonstrated in this paper that such a simplicial-based construction provides a human-understandable explanation of the decision.

The ideas presented in this paper can be extended in many different ways. In future work, we intend to study less-data-dependent approaches so that the Delaunay triangulation is needless. Besides, this architecture should be extended to Deep Learning models so that it can be applied to more complex classification problems such as image classification and provides extra explainability to Deep Learning models.

Code availability

The code is available in the GitHub repository: https://github.com/Cimagroup/TrainableSMNN.

CRediT authorship contribution statement

Eduardo Paluzo-Hidalgo: Conceptualization, Formal analysis, Investigation, Methodology, Software, Validation, Visualization, Writing – original draft, Writing – review & editing. **Rocio Gonzalez-Diaz:** Conceptualization, Formal analysis, Funding acquisition, Investigation, Methodology, Project administration, Supervision, Writing – original draft, Writing – review & editing. **Miguel A.**

Gutiérrez-Naranjo: Conceptualization, Formal analysis, Investigation, Methodology, Software, Supervision, Writing – original draft, Writing – review & editing.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Data availability

Data will be made available on request.

Acknowledgements

References

[1] M. Graziani, L. Dutkiewicz, D. Calvaresi, A global taxonomy of interpretable AI: unifying the terminology for the technical and social sciences, Artif. Intell. Rev. (2022), https://doi.org/10.1007/s10462-022-10256-8.

[2] C. Molnar, Interpretable Machine Learning, 2 ed., independently published, 2022, https://christophm.github.io/interpretable-ml-book.

[3] W.J. Murdoch, C. Singh, K. Kumbier, et al., Definitions, methods, and applications in interpretable machine learning, Proc. Natl. Acad. Sci. (2019), https://www.pnas.org/doi/full/10.1073/pnas.1900654116.

[4] E. Paluzo-Hidalgo, R. Gonzalez-Diaz, M.A. Gutiérrez-Naranjo, Two-hidden-layer feed-forward networks are universal approximators: a constructive approach, Neural Netw. 131 (2020) 29–36, https://doi.org/10.1016/j.neunet.2020.07.021.

[5] E. Paluzo-Hidalgo, R. Gonzalez-Diaz, M.A. Gutiérrez-Naranjo, J. Heras, Simplicial-map neural networks robust to adversarial examples, Mathematics 9 (2021) 169, https://doi.org/10.3390/math9020169.

[6] E. Paluzo-Hidalgo, R. Gonzalez-Diaz, M.A. Gutiérrez-Naranjo, J. Heras, Optimizing the simplicial-map neural network architecture, J. Imaging 7 (2021) 173, https://doi.org/10.3390/jimaging7090173.

[7] N.J. Guliyev, V. Ismailov, On the approximation by single hidden layer feedforward neural networks with fixed weights, Neural Netw. 98 (2018) 296–304, https://doi.org/10.1016/j.neunet.2017.12.007.

[8] H. Edelsbrunner, J. Harer, Computational Topology - an Introduction, Am Math Soc, 2010.

[9] J. Boissonnat, F. Chazal, M. Yvinec, Geometric and Topological Inference, Cambridge Texts in Applied Maths, Cambridge Univ Press, 2018.

[10] L. McInnes, J. Healy, J. Melville, UMAP: uniform manifold approximation and projection for dimension reduction, http://arxiv.org/abs/1802.03426, 2018.

[11] P.P. Das, L. Wiese, Explainability based on feature importance for better comprehension of machine learning in healthcare, in: A. Abelló, P. Vassiliadis, O. Romero, R. Wrembel, F. Bugiotti, J. Gamper, G. Vargas Solar, E. Zumpano (Eds.), New Trends in Database and Information Systems, Springer Nature, Switzerland, Cham, 2023, pp. 324–335.

[12] A. Nguyen, J. Yosinski, J. Clune, Understanding Neural Networks via Feature Visualization: A Survey, Springer International Publishing, Cham, 2019, pp. 55–76.

[13] Z. Kong, K. Chaudhuri, Understanding instance-based interpretability of variational auto-encoders, in: A. Beygelzimer, Y. Dauphin, P. Liang, J.W. Vaughan (Eds.), Advances in Neural Information Processing Systems, 2021, https://openreview.net/forum?id=a5-37ER8qTI.

[14] I.R. Subramanian, Design of experiments for the nips 2003 variable selection benchmark, in: NeurIPS Proceedings, 2003.

[15] E. Paluzo-Hidalgo, R. Gonzalez-Diaz, M.A. Gutiérrez-Naranjo, Topology-based representative datasets to reduce neural network training resources, Neural Comput. Appl. 34 (2022) 14397–14413, https://doi.org/10.1007/s00521-022-07252-y.

[16] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, E. Duchesnay, Scikit-learn: machine learning in Python, J. Mach. Learn. Res. 12 (2011) 2825–2830.

[17] D. Kingma, J. Ba, Adam: a method for stochastic optimization, in: Y. Bengio, Y. LeCun (Eds.), 3rd Int Conf on Learning Representations, ICLR 2015, Conference Track Proceedings, San Diego, CA, USA, May 7-9, 2015, 2015, http://arxiv.org/abs/1412.6980.

[18] B.J. Stolz, Outlier-robust subsampling techniques for persistent homology, J. Mach. Learn. Res. 24 (2023) 1–35, http://jmlr.org/papers/v24/21-1526.html.