Topology Tools for Explainable and Green Artificial Intelligence

Rocio Gonzalez-Diaz
rogodi@us.es

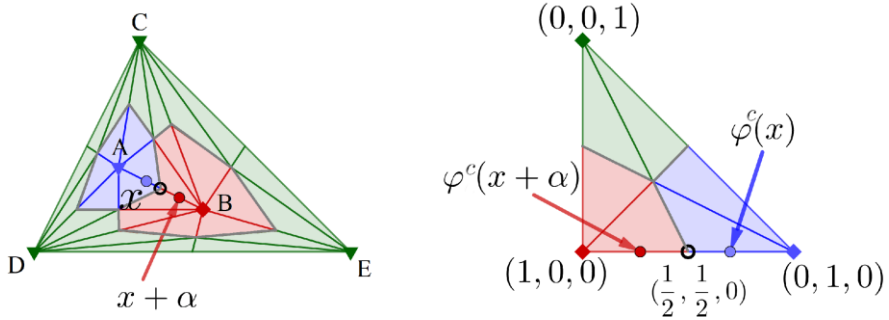
Sixth EACA International School on Computer Algebra and its Applications

SANTIAGO DE COMPOSTELA, JULY 18-21, 2023

- Context: Green and Explainable artificial intelligence (REXASI-PRO)
- Computational topology tools: Persistent homology, barcodes, distance bottleneck, simplicial maps, Persistence modules, morphisms between persistence modules
- Partial matchings between barcodes
- Simplicial maps neural networks

Topology Tools for Explainable and Green Artificial Intelligence

Rocio Gonzalez-Diaz
rogodi@us.es



- Context: Green and Explainable artificial intelligence (REXASI-PRO)
- Computational topology tools: Persistent homology, barcodes, distance bottleneck, simplicial maps, Persistence modules, morphisms between persistence modules
- Partial matchings between barcodes
- Simplicial maps neural networks

Simplicial maps neural networks

Joint work with **Eduardo Paluzo-Hidalgo** and **Miguel Ángel Gutiérrez-Naranjo**

Eduardo Paluzo-Hidalgo

Assistant Professor

epaluzo@us.es

Miguel-Angel Gutierrez-Naranjo

Associate Professor

magutier@us.es

Simplicial maps neural networks

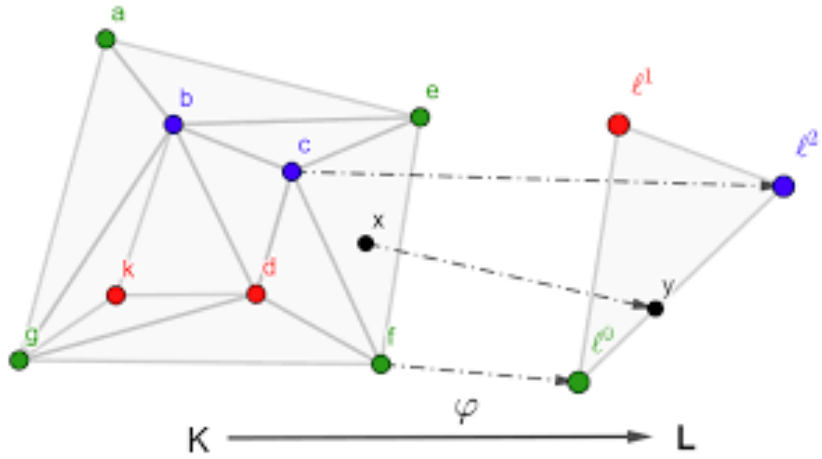
Recall:

Given two simplicial complexes K and L, a vertex map $\varphi^{(0)} : K^{(0)} \to L^{(0)}$ is a function from the vertices of K to the vertices of L such that for any simplex $\sigma \in K$, the set $\varphi(\sigma) := \{v \in L^{(0)} \ : \ \exists u \in \sigma, \ \varphi^{(0)}(u) = v\}$ is a simplex of L.

The simplicial map $\varphi: |K| \to |L|$ induced by the vertex map $\varphi^{(0)}$ is a continuous function defined as

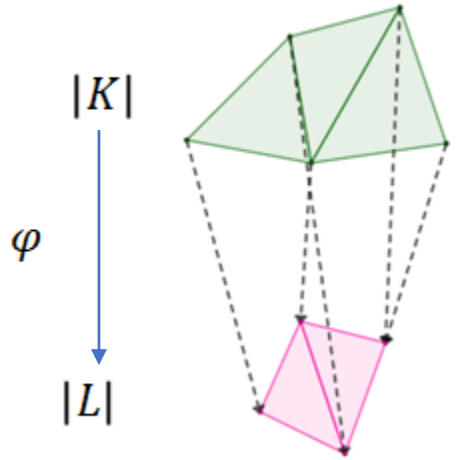
$$\varphi(x) = \sum_{i=0}^{n} b_i(x)\varphi^{(0)}(u_i)$$

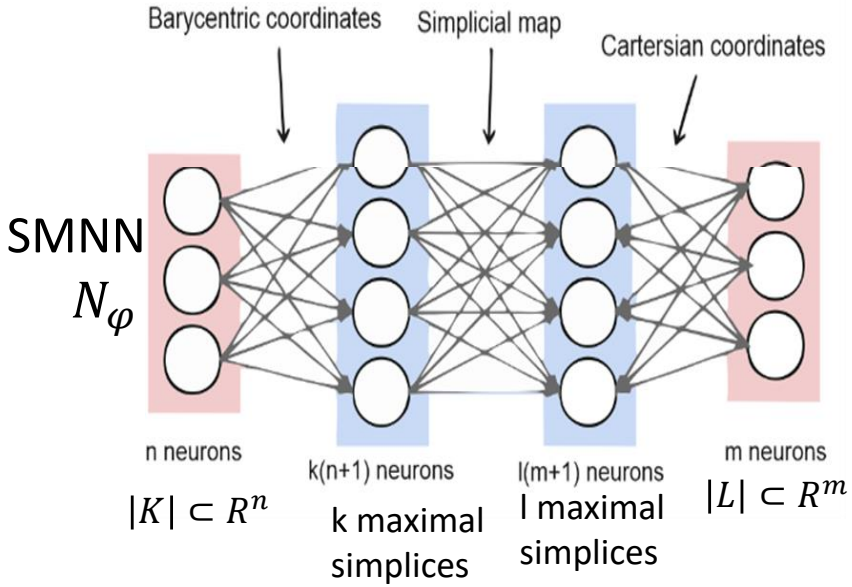
where $(b_0(x), \dots, b_k(x))$ are the barycentric coordinates of x wrt $\sigma = \langle u_0, \dots, u_n \rangle \in K$ with $x \in \sigma$.

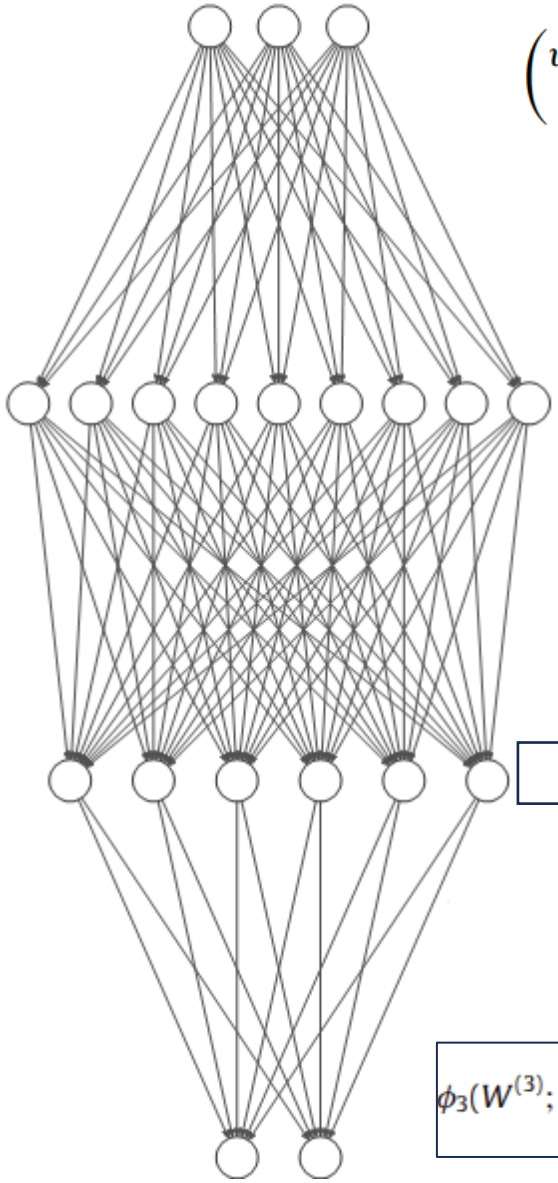


Simplicial maps neural networks

Weights are defined by a simplicial map between triangulations of the input and output spaces.







$$\begin{pmatrix} v_0^i & \cdots & v_n^i \\ 1 & \cdots & 1 \end{pmatrix}^{-1} = (W_i^{(1)} \mid B_i)$$

$$W^{(1)} = \begin{pmatrix} W_1^{(1)} \\ \vdots \\ W_k^{(1)} \end{pmatrix} \quad b_1 = \begin{pmatrix} B_1 \\ \vdots \\ B_k \end{pmatrix}$$

$$\phi_1(W^{(1)}; y; b_1) := W^{(1)} y + b_1$$

$$W^{(2)} = (W^{(2)}_{s_1,s_2})$$

where

$$W^{(2)}_{s_1,s_2} = \begin{cases} 1 \text{ if } \varphi(v_t^i) = u_r^j, \\ 0 \text{ otherwise;} \end{cases}$$

being $s_1 = j(r+1)$ and $s_2 = i(t+1)$

$$\phi_2(W^{(2)}; y; b_2) := W^{(2)} y$$

$$W_j^{(3)} = (u_0^j \quad \cdots \quad u_m^j)$$

$$\phi_3(W^{(3)}; y; b_3) = \frac{\sum_{j=1}^{\ell} W_j^{(3)} y^j \, \psi(y^j)}{\sum_{j=1}^{\ell} \psi(y^j)}$$

Drawbacks:

- Not trainable
- Very expensive!!

https://doi.org/10.1016/j.neunet.2020.07.021

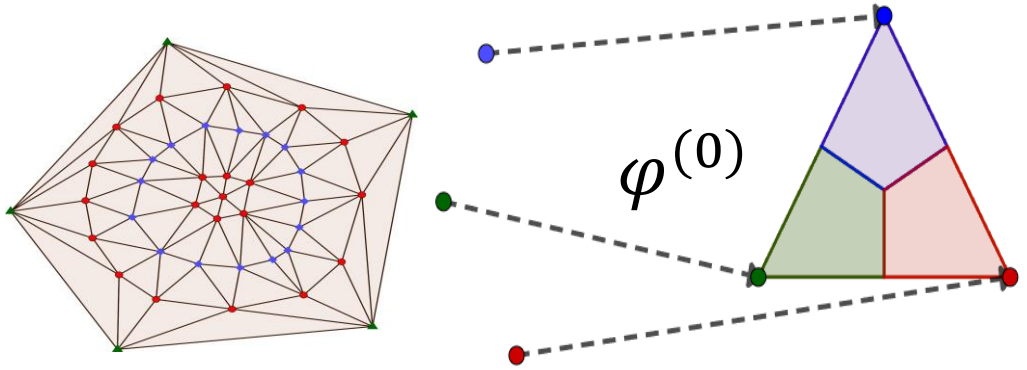
Simplicial maps neural networks

Classification using SMNNs:

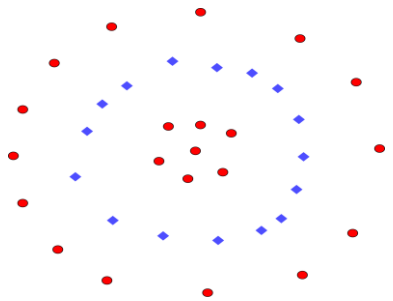
1. P convex polygon surrounding data.
2. Delaunay triangulation of data and the vértices of P.
3. Maximal simplex encoding the labels.
4. Simplicial map φ where $\varphi^{(0)}(v)$ is the one-hot encoding of the class of v.
5. SMNN N_φ

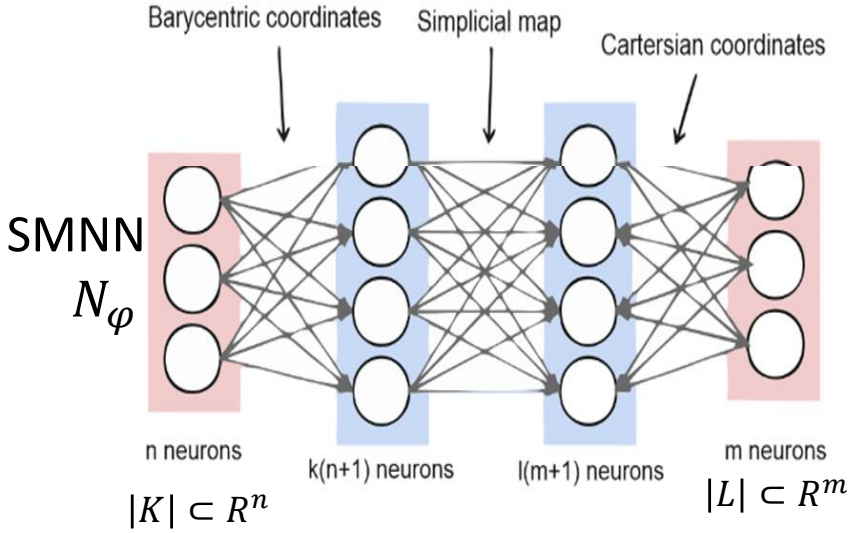
Simplicial map

$|K| \xrightarrow{\varphi} |L|$



Input data





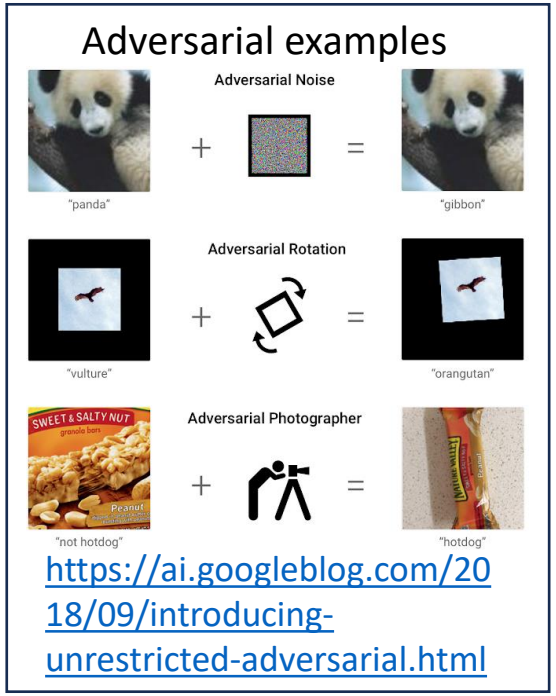
Drawbacks:

- Not robust against adversarial examples
- Not trainable
- Very expensive!!
- Polygon surrounding the data

https://doi.org/10.3390/math9020169

Simplicial maps neural networks

SMNNs robust to adversarial examples:



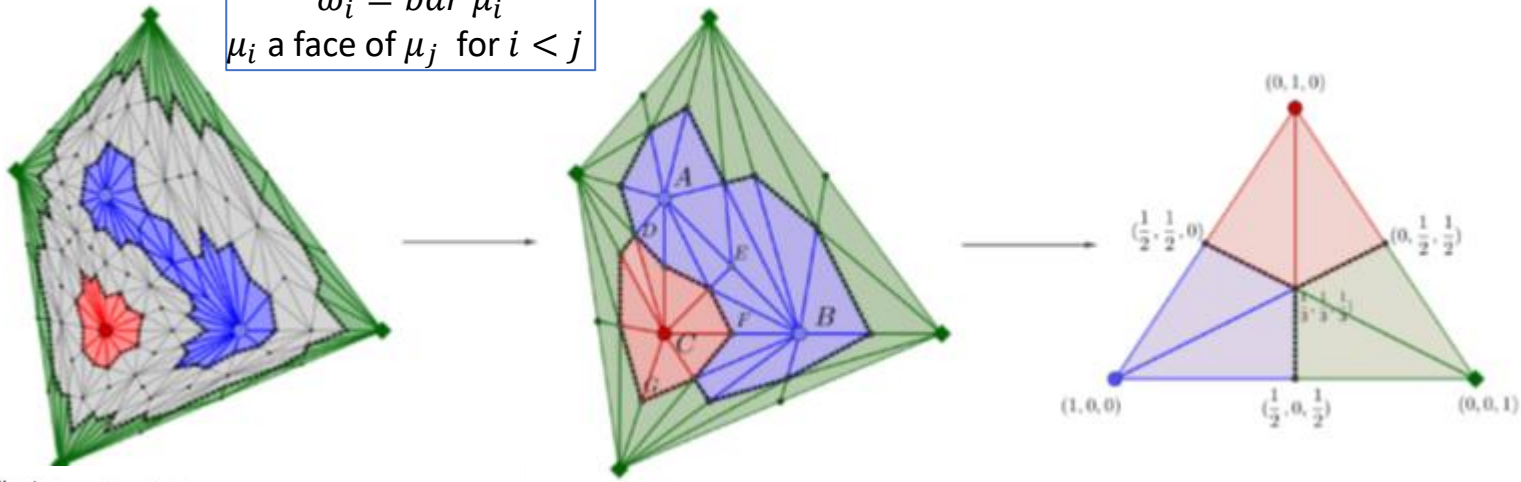
https://ai.googleblog.com/2018/09/introducing-unrestricted-adversarial.html

$\sigma \in SdK$

$\sigma = \langle \omega_0, \dots, \omega_k \rangle$

$\omega_i = bar\ \mu_i$

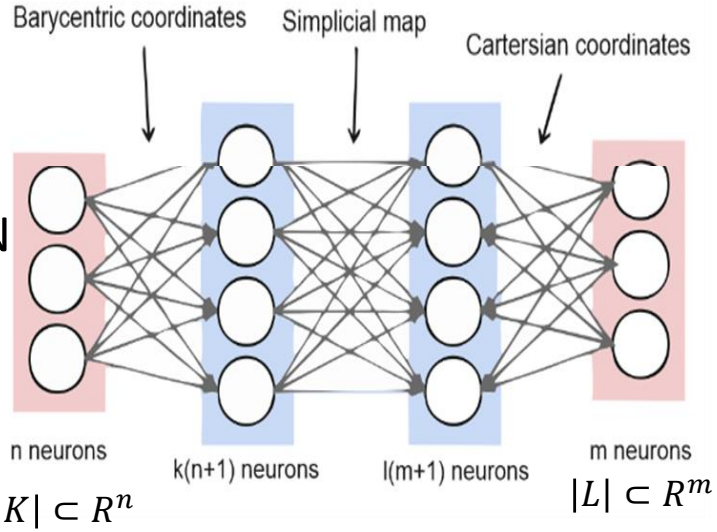
μ_i a face of μ_j for $i < j$



$$w = \text{bar}\,\mu$$

$$\varphi_t^{(0)}(w) := \text{bar}\,\varphi_{t-1}(\mu)$$

SMNN N_φ



$|K| \subset R^n$ $|L| \subset R^m$

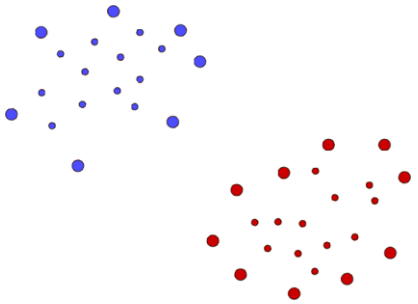
Drawbacks:

- Not trainable
- Still very expensive!!
- Polygon surrounding the data

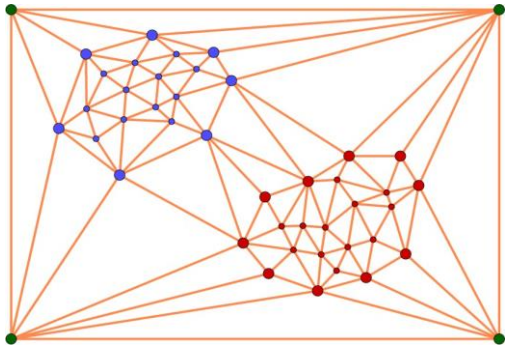
https://doi.org/10.3390/math9020169

Simplicial maps neural networks

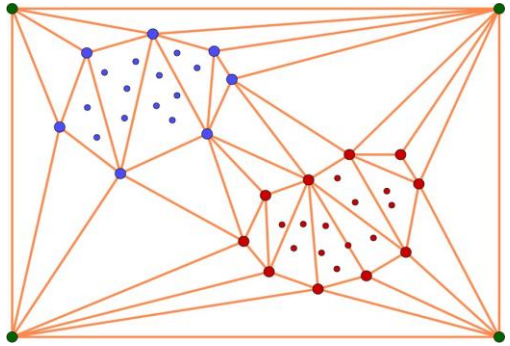
Optimizing SMNNs:



N_φ

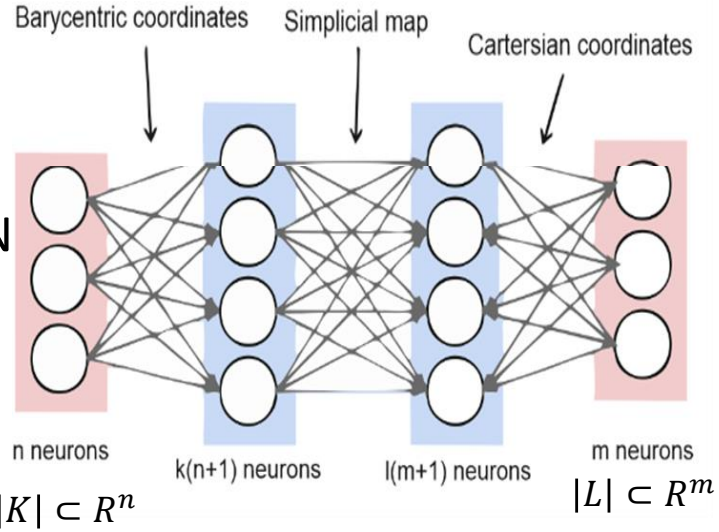


$N_{\tilde{\varphi}}$



Both N_φ and $N_{\tilde{\varphi}}$ correctly classify P

SMNN N_φ



Drawbacks:

- Not trainable
- Still very expensive!!
- Polygon surrounding the data

https://doi.org/10.3390/jimaging7090173

Simplicial maps neural networks

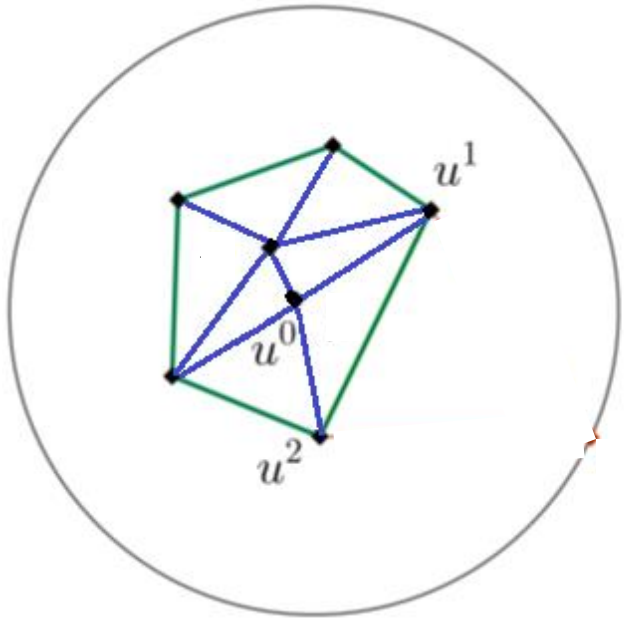
Training SMNNs. Preprocessing

1. Select $U \subset V$. Compute $Del(U)$.
2. Compute the *boundary* of the *known* space.
3. Compute the barycentric coordinates of V wrt $Del(U)$

Input data:
$V = \{v\}$.
k labels

$U = \{u^1, \dots, u^\alpha\}$

center_of_$mass(U)$ = origin o.
Fix $r \in R$ such that $V \subset B^n(o, r)$.



For v $\in V$. If $v \in Del(U)$:
compute $\sigma = \langle w^{i_0}, \dots, w^{i_n} \rangle \in Del(U)$ such that v $\in \sigma$.

For v $\in V$. If $v \notin Del(U)$:

Compute $\Gamma = \{ \mu \in \delta Del(U) : (N \cdot u^{i_0} + c)(N \cdot v + c) < 0 \}$

Compute $w = r \frac{v}{||v||} \in S^n(o, r)$.

$\mu = \langle u^{i_1}, \dots, u^{i_n} \rangle$
$\mu \subset \langle u^{i_0}, u^{i_1}, \dots, u^{i_n} \rangle \in$ K

Find $\sigma = \langle w, u^{i_1}, \dots, u^{i_n} \rangle$ such that $\langle u^{i_1}, \dots, u^{i_n} \rangle \in \Gamma$ and $v \in |\sigma|$.

https://arxiv.org/abs/2306.00010

Simplicial maps neural networks

Training SMNNs. Preprocessing

1. Select $U \subset V$. Compute $Del(U)$.
2. Compute the *boundary* of the *known* space.
3. Compute the barycentric coordinates of V wrt $Del(U)$

Input data:
$V = \{v\}$.
k labels

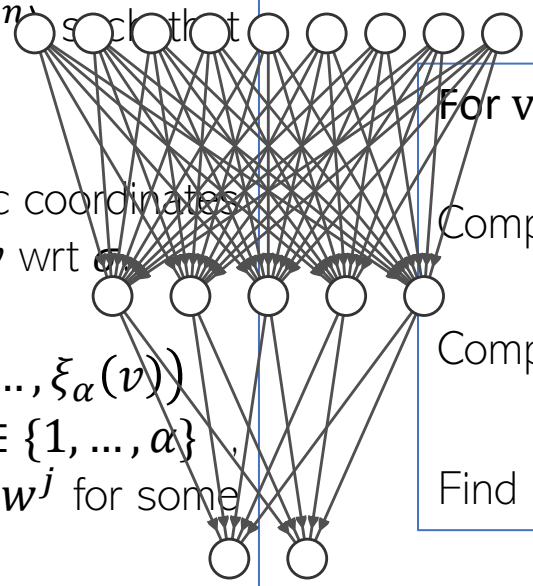
$U = \{u^1, \dots, u^\alpha\}$

center_of_$mass(U)$ = origin o.
Fix $r \in R$ such that $V \subset B^n(o, r)$.

For $\mathbf{v} \in \boldsymbol{V}$:
Let $\sigma = \langle w^0, w^1, \dots, w^n \rangle$ such that $v \in \sigma$.

Compute the baricentric coordinates $(b_o(v), \dots, b_n(v))$ of v wrt σ.

Then, $\xi(v) := (\xi_1(v), \dots, \xi_\alpha(v))$ such that, for $t \in \{1, \dots, \alpha\}$, $\xi_t(x) = b_j(x)$ if $u^t = w^j$ for some $j \in \{0, \dots, n\}$.



For $\mathrm{v} \in V$. If $v \in Del(U)$:
compute $\sigma = \langle w^{i_0}, \dots, w^{i_n} \rangle \in Del(U)$ such that $\mathrm{v} \in \sigma$.

For $\mathrm{v} \in V$. If $v \notin Del(U)$:

Compute $\Gamma = \{ \mu \in \delta Del(U) : (N \cdot u^{i_0} + c)(N \cdot v + c) < 0 \}$

$\mu = \langle u^{i_1}, \dots, u^{i_n} \rangle$
$\mu \subset \langle u^{i_0}, u^{i_1}, \dots, u^{i_n} \rangle \in \mathrm{K}$

Compute $w = r \frac{v}{||v||} \in S^n(o, r)$.

Find $\sigma = \langle w, u^{i_1}, \dots, u^{i_n} \rangle$ such that $\langle u^{i_1}, \dots, u^{i_n} \rangle \in \Gamma$ and $v \in |\sigma|$.

https://arxiv.org/abs/2306.00010

Simplicial maps neural networks

Training SMNNs. Initialization

1. Input data: $\{\xi(v): v \in V\}$
2. For $u \in U$: assign random vectors $\varphi_U^{(0)}(u) \in R^k$
3. For $\mathbf{v} \in V$: compute $\varphi_U(v)$

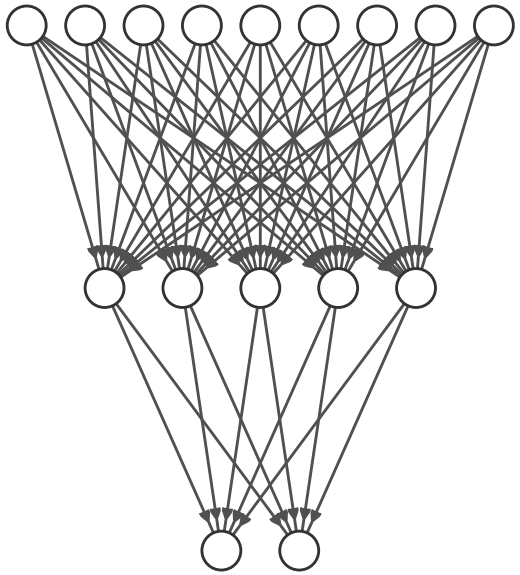
Input data:
$V = \{v\}$.
k labels

$U = \{u^1, \dots, u^\alpha\}$

For $\mathbf{v} \in \boldsymbol{V}$: $\xi(v) := (\xi_1(v), \dots, \xi_\alpha(v))$

$\varphi_U^{(0)}(u^t) = (p_1^t, \dots, p_k^t)$ for $u^t \in U$ and $t \in \{1, \dots, \alpha\}$

$$\varphi_U(v) := softmax\left(\sum_{t=1}^{\alpha} \xi_t(v)\varphi_U^{(0)}(u^t)\right)$$



https://arxiv.org/abs/2306.00010

Simplicial maps neural networks

Training SMNNs. Learning

1. Using gradient descent:

$$p_j^t := p_j^t - \eta \frac{\partial \mathcal{L}(\varphi_U, \varphi^{(0)}, v)}{\partial p_j^t} = p_j^t - \eta (s_j - y_j)\xi_t(v)$$

Input data:
$V = \{v\}$.
k labels

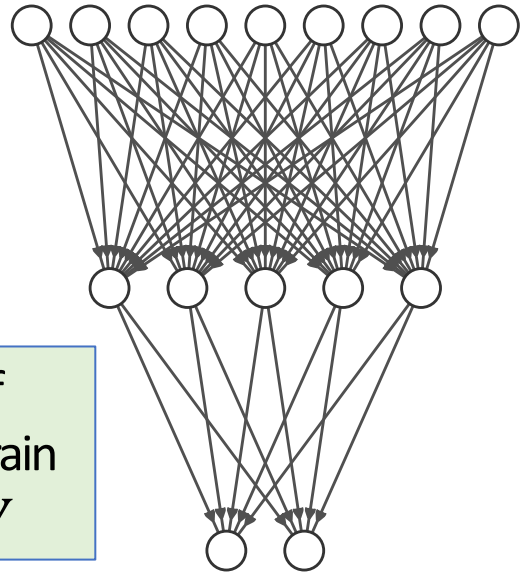
$$U = \{u^1, \dots, u^\alpha\}$$

For $\mathbf{v} \in \boldsymbol{V}$: $\xi(v) := (\xi_1(v), \dots, \xi_\alpha(v))$

$\varphi_U^{(0)}(u^t) = (p_1^t, \dots, p_k^t)$ for $u^t \in U$ and $t \in \{1, \dots, \alpha\}$

$$\varphi_U(v) := softmax\left(\sum_{t=1}^{\alpha} \xi_t(v)\varphi_U^{(0)}(u^t)\right)$$

$$\eta = learning\_rate$$



If U $\notin V$, then U is a set of synthetic samples that can train the model quicker than V

The error function (categorical crossentropy):

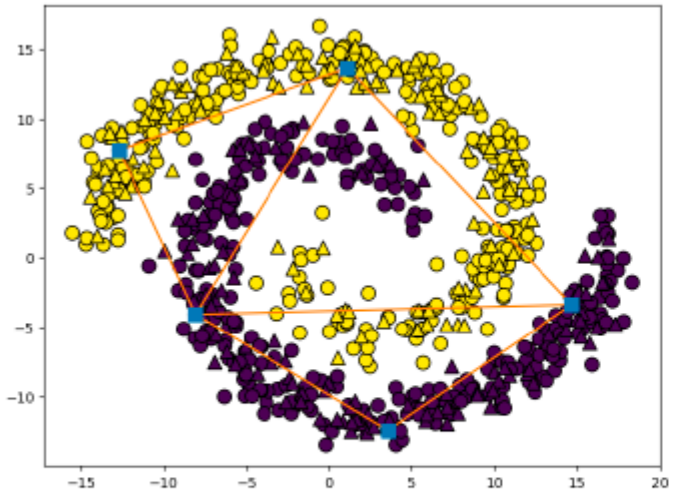
$$\mathcal{L}(\varphi_U, \varphi^{(0)}, v) = -\sum_{h=1}^{k} y_h \log(s_h)$$

where $(y_1, \dots, y_k) = \varphi^{(0)}(v)$ is the one-hot encoding of the class of v and $(s_1, \dots, s_k) = \varphi_U(v)$.

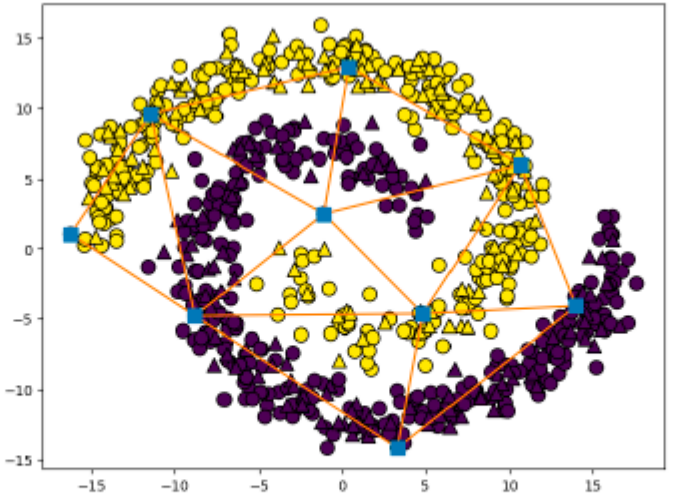
https://arxiv.org/abs/2306.00010

Simplicial maps neural networks

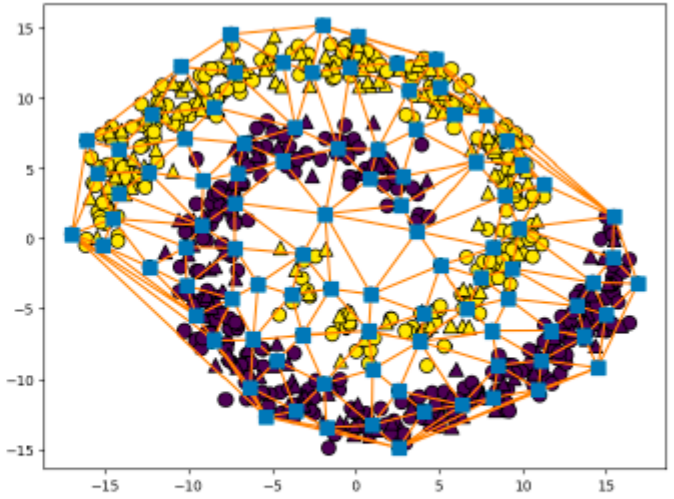
Experiments $\boldsymbol{U} \subset \boldsymbol{V}$



(a) Using 5 support points. The SMNN reaches 80% of accuracy on the test set.



(b) Using 9 support points. The SMNN reaches 93% of accuracy on the test set.

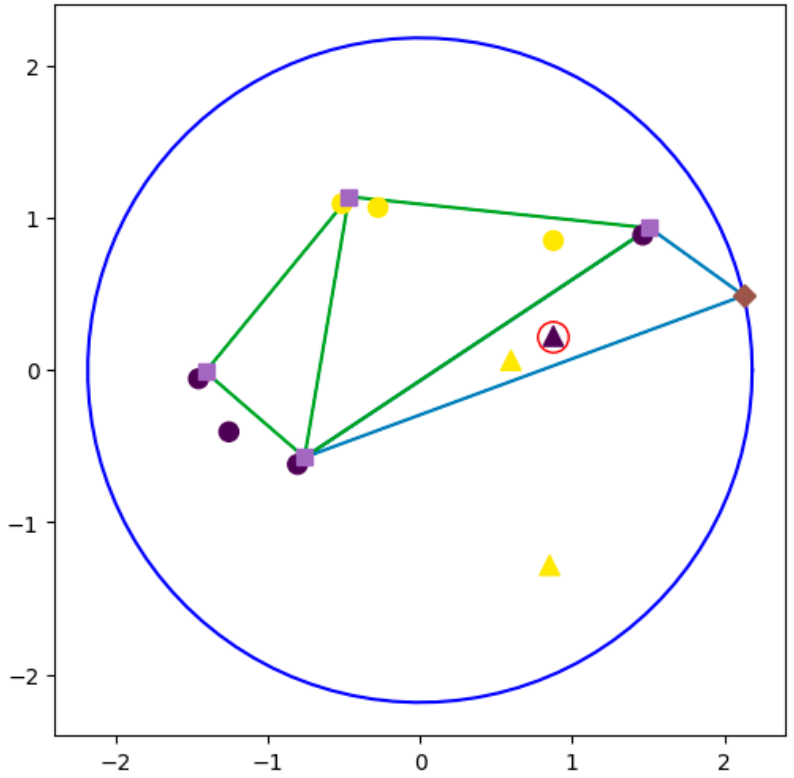


(c) Using 95 support points. The SMNN reaches 99% of accuracy on the test set.

https://arxiv.org/abs/2306.00010

Simplicial maps neural networks

Experiments $\boldsymbol{U} \subset \boldsymbol{V}$



	SMNN				NN	
n	ε	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		

https://arxiv.org/abs/2306.00010

Links

https://arxiv.org/abs/2306.00010

https://github.com/Cimagroup/TrainableSMNN

Topology Tools for Explainable and Green Artificial Intelligence

Rocio Gonzalez-Diaz
rogodi@us.es

- Context: Green and Explainable artificial intelligence (REXASI-PRO)
- Computational topology tools: Persistent homology, barcodes, distance bottleneck, simplicial maps, Persistence modules, morphisms between persistence modules
- Partial matchings between barcodes
- Simplicial maps neural networks