A neural approach to point membership classification using local boundary samples

Bruno Degli Esposti[1], Giuseppe Alessio D'Inverno[2], Francesca Pelosi[1], and Maria Lucia Sampoli[1]

[1]Dep. of Information Engineering and Mathematics, University of Siena

[2]Laboratoire de Mathematiques d'Orsay, Université Paris-Saclay

UNIVERSITÀ
DI SIENA
1240

Young Applied Mathematicians Conference, 6th edition

17 September 2026, Torino, Italy

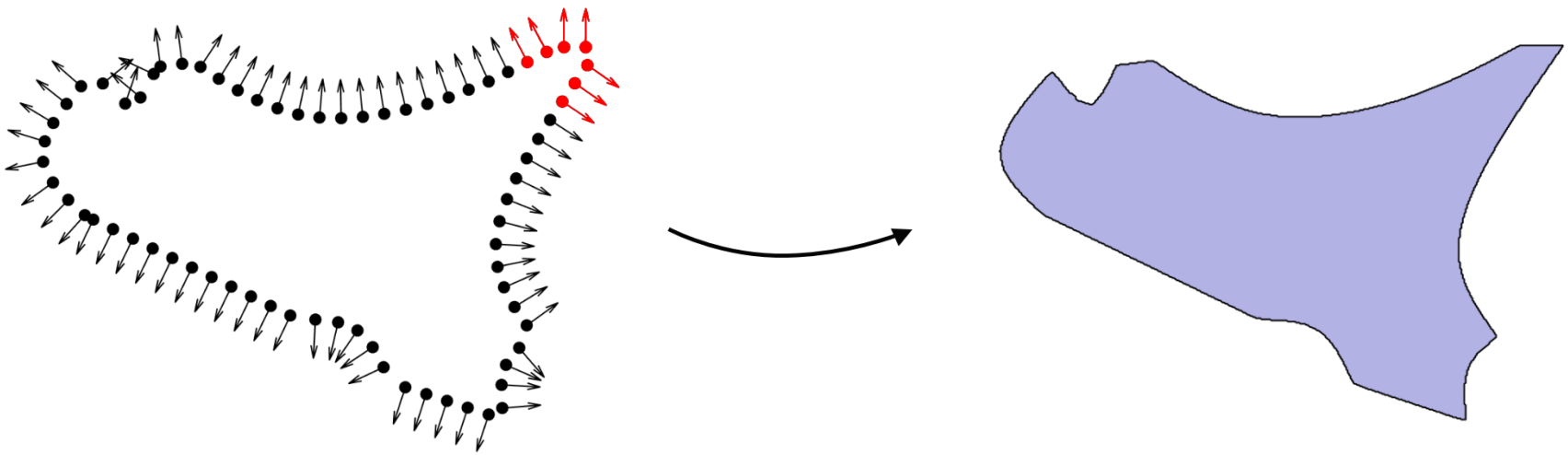
Introduction

Introduction

Topic of this talk

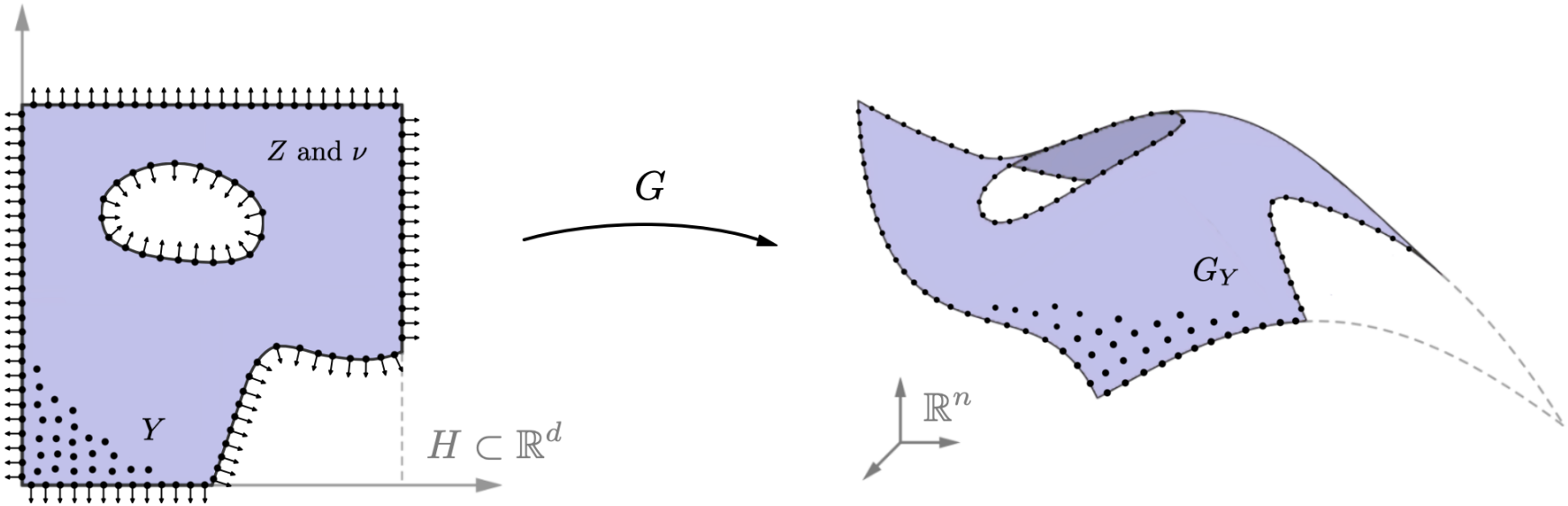
Given unstructured points and normals on the boundary of a domain, how can we approximately test for inclusion in the domain?

Our approach: computational geometry + small neural network.
We extract small local sets of boundary points close to the query point to test for inclusion. Key advantages: speed and domain independence.



Target application for this work

Advancing front point cloud generation is an alternative to mesh generation for meshless workflows (RBF, kernels, etc). Each point spawns more points around it and the domain is eventually filled up in a uniform way. New points must be quickly tested for inclusion!



Advancing front point cloud generation

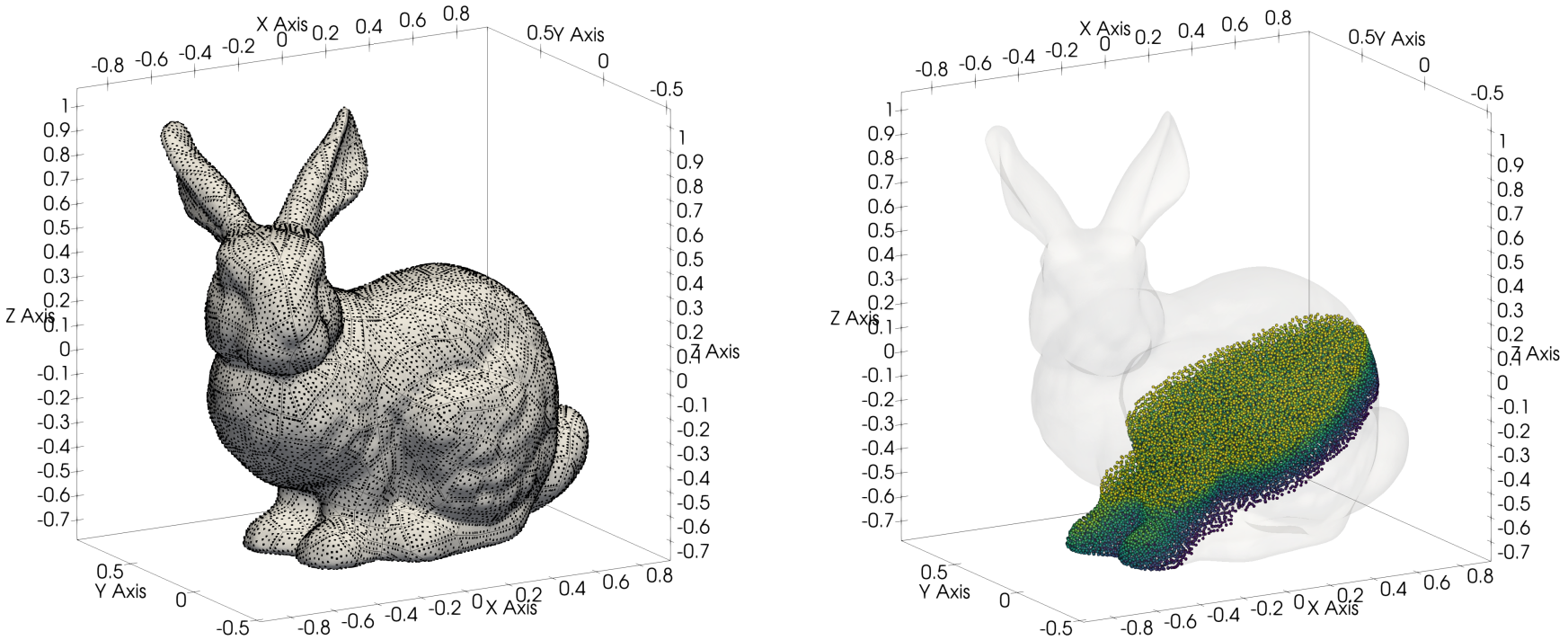


Figure: Points generated by surface and volume advancing front algorithms. Our inclusion tests successfully restrict volume point cloud generation to the bunny interior using only boundary nodes and normals.

Point membership classification using boundary samples

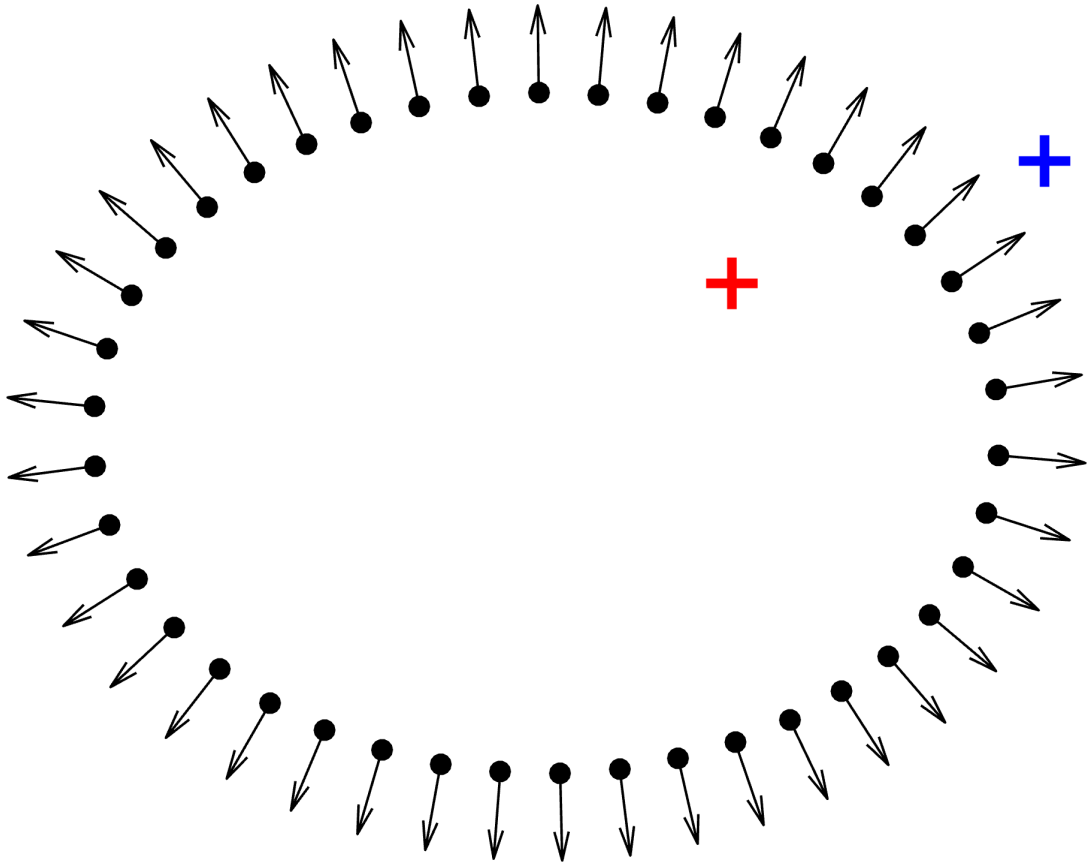


Figure: Boundary points samples $z_i \in Z$ with normals $\nu_i \in \nu$. Query points y_R in red and y_B in blue. Which one is inside?

Point membership classification using boundary samples

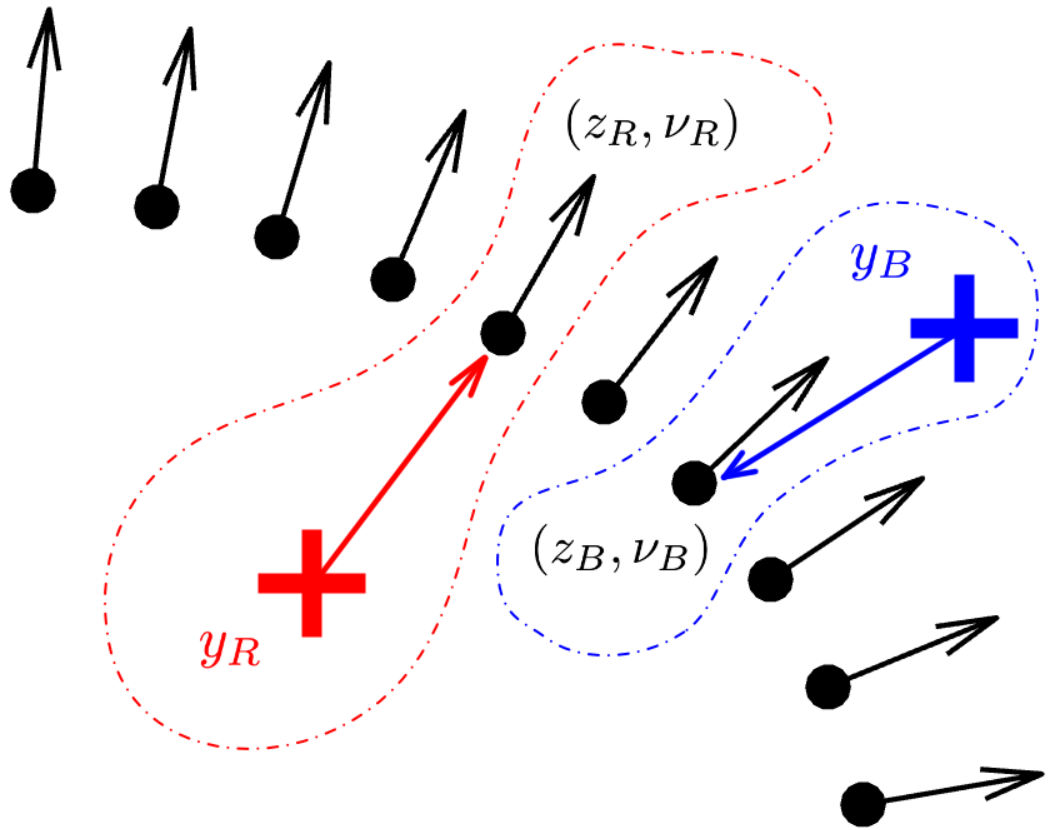


Figure: z_R and z_B are the boundary samples closest to y_R and y_B. Note that $(z_R - y_R) \cdot \nu_R > 0$ and $(z_B - y_B) \cdot \nu_B < 0$.

Point membership classification using boundary samples

Approximate inclusion test INN1

Input: Point $y \in \mathbb{R}^d$ to be tested for inclusion in a domain Ω.

Input: Set of boundary points $Z \subset \partial\Omega$ and corresponding set of unit vectors ν normal to $\partial\Omega$ and pointing outwards with respect to Ω.

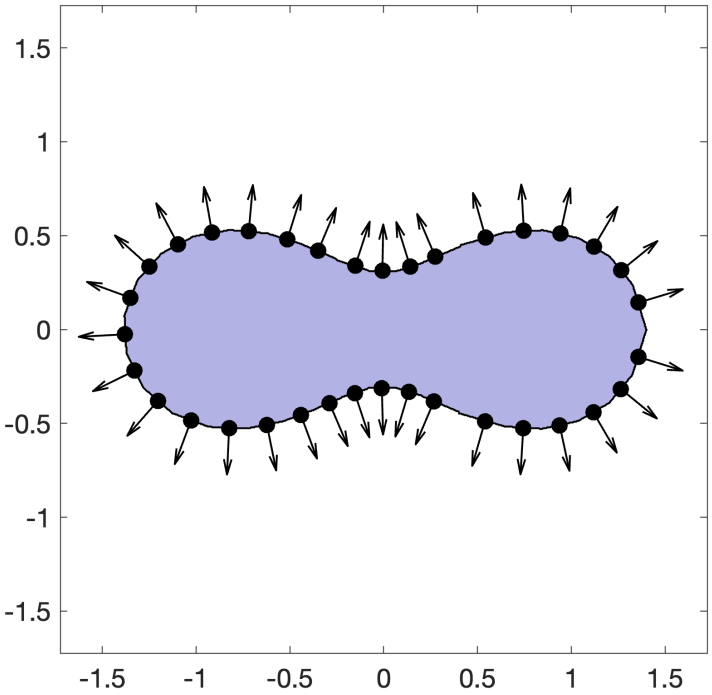
Output: Boolean result of the inclusion test.

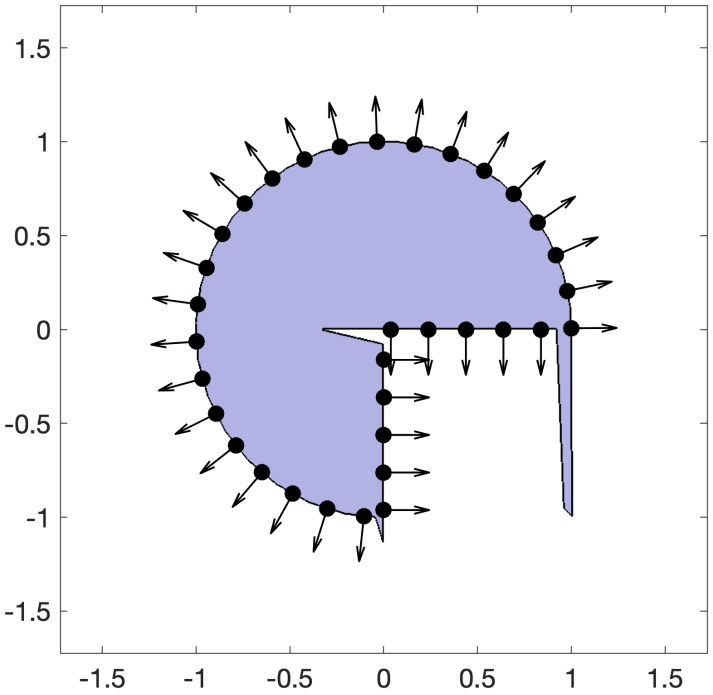
1: Find the index i of the boundary point in Z closest to y.
2: **if** $(z_i - y) \cdot \nu_i > 0$ **then**
3: **return** true
4: **else**
5: **return** false
6: **end if**

Runtime: $\mathcal{O}(\log N)$ for N boundary points, with the aid of a k-d tree.

Point membership classification using boundary samples

The half-plane test INN1 has unbeatable performance. How reliable is it for complex domains? We color in violet the region where it returns true, which we call reconstructed domain.





Robust around holes and nonconvex features. Fragile around corners!

Order of convergence of inclusion tests

Can we prove convergence for smooth bounded domains? How do we even define convergence for an inclusion test relying on boundary nodes?

Definition

For any $Z \subset \partial\Omega$, we define its intrinsic fill distance as

$$h_Z = \sup_{z \in \partial\Omega} \min_{z' \in Z} d(z, z')$$

with $d(z, z')$ being the length of the shortest curve connecting z to z'.

Definition

An inclusion test has order of convergence $k \geq 0$ for a bounded Ω if there exists $C > 0$ such that $\forall Z \subset \partial\Omega$ the test is exact for any query point $y \in \mathbb{R}^d$ whose distance from $\partial\Omega$ is greater than Ch_Z^k.

Order of convergence of inclusion tests

Proposition (smooth case)

The test INN1 has order 2 for any planar C^2 bounded domain.

Open problem

How to extend the proof to C^2 bounded domains in $\mathbb{R}^d$ with $d > 2$?

Proposition (nonsmooth case)

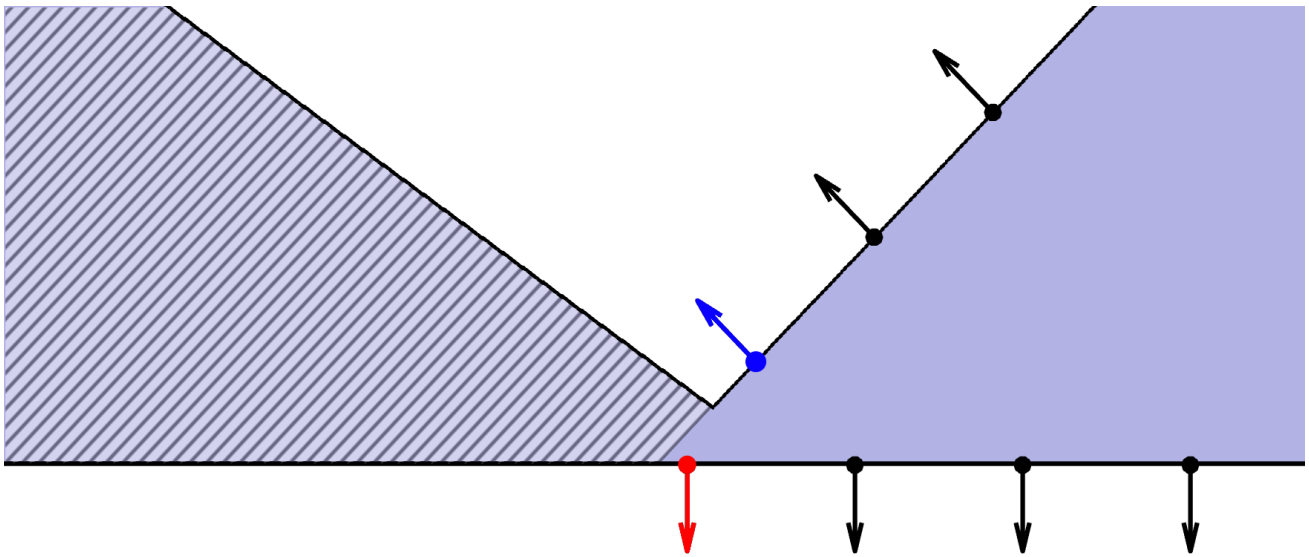
INN1 does not even have order 0 on piecewise smooth domains.

Remark

For reference, triangularizing a piecewise C^2 bounded domain and testing for inclusion in the mesh gives a method of order 2.

Issues around corners

How does INN1 fail around corners, and when? Example in 2D:



The test incorrectly returns true for all points in the shaded region, because the red boundary sample is the closest. Idea: use $K > 1$ nearest boundary points. With $K = 2$ we get access to the blue boundary sample which correctly excludes the shaded region.

Using K nearest boundary samples

Using K nearest boundary samples

Approximate inclusion test INNK-all

Input: Point $y \in \mathbb{R}^d$ to be tested for inclusion in a domain Ω.
Input: Set of boundary points $Z \subset \partial\Omega$ and corresponding set of unit vectors ν normal to $\partial\Omega$ and pointing outwards with respect to Ω.
Input: Number K of nearest boundary samples to search.

1: Find indices $i_1, \ldots, i_K$ of the K boundary points in Z closest to y
2: **for** $i \in \{i_1, \ldots, i_K\}$ **do**
3: **if** $(z_i - y) \cdot \nu_i \leq 0$ **then**
4: **return** false
5: **end if**
6: **end for**
7: **return** true

Problem solved?

Using K nearest boundary samples

Approximate inclusion test INNK-all

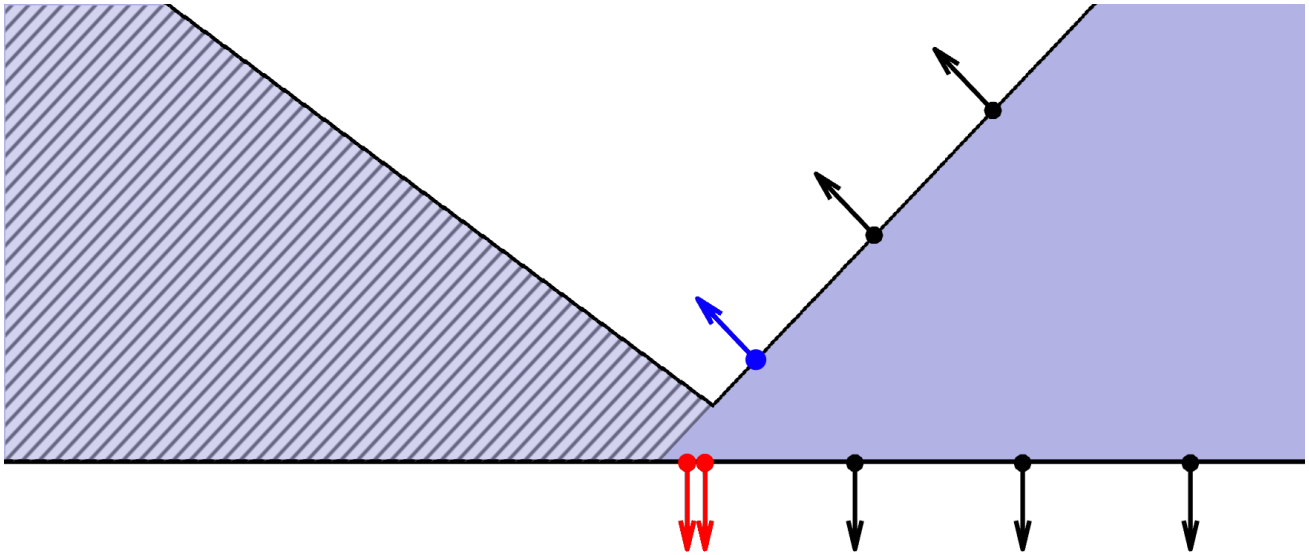
Input: Point $y \in \mathbb{R}^d$ to be tested for inclusion in a domain Ω.
Input: Set of boundary points $Z \subset \partial\Omega$ and corresponding set of unit vectors ν normal to $\partial\Omega$ and pointing outwards with respect to Ω.
Input: Number K of nearest boundary samples to search.

1: Find indices $i_1, \ldots, i_K$ of the K boundary points in Z closest to y
2: **for** $i \in \{i_1, \ldots, i_K\}$ **do**
3: **if** $(z_i - y) \cdot \nu_i \leq 0$ **then**
4: **return** false
5: **end if**
6: **end for**
7: **return** true

Problem solved? Not really. Three issues: two trivial, one serious.

Clustered boundary samples

Clusters of boundary points can recreate the same artifacts of the naive inclusion test INN1. For example, with $K = 2$,



and the issue generalizes to any $K > 1$. Easy fix: discard some boundary samples to enforce minimum spacing between points.

Clustered boundary samples

Definition

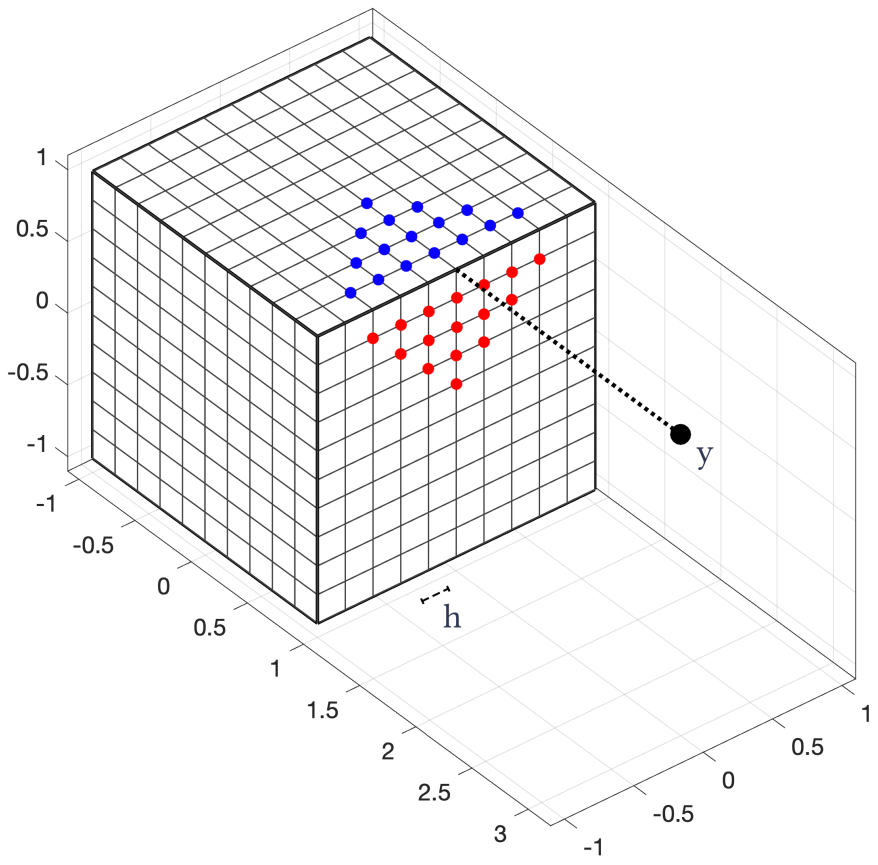
For any $Z \subset \partial\Omega$, we define its separation distance as

$$q_Z = \frac{1}{2} \min_{z,z' \in Z,\, z \neq z'} d(z, z').$$

Note that the minimum K for a meaningful inclusion test on a piecewise smooth domain depends linearly on the ratio h_Z/q_Z, and exponentially on the dimension of the domain.

My use case: advancing front method generates points on $\partial\Omega$ with $h_Z/q_Z \approx 1$, so $K = 2$ in 2D and $K \approx 5$ in 3D are sufficient.
We keep this assumption for the rest of the talk.

Switching to local sets of boundary samples



In the example, it follows from

$$\sqrt{D^2 + h^2} = D + \mathcal{O}(h^2)$$

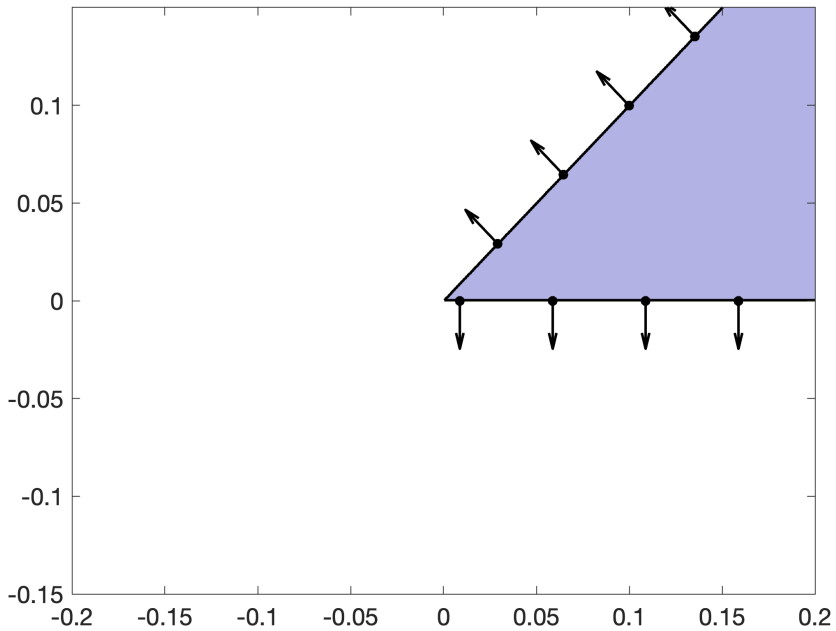
that red points are at distance $D + \mathcal{O}(h^2)$ from y, while blue points are at distance $D + \mathcal{O}(h)$.

The K nearest boundary points will all belong to the "red" face for small h! However, we need points from both faces to detect the cube's edge.

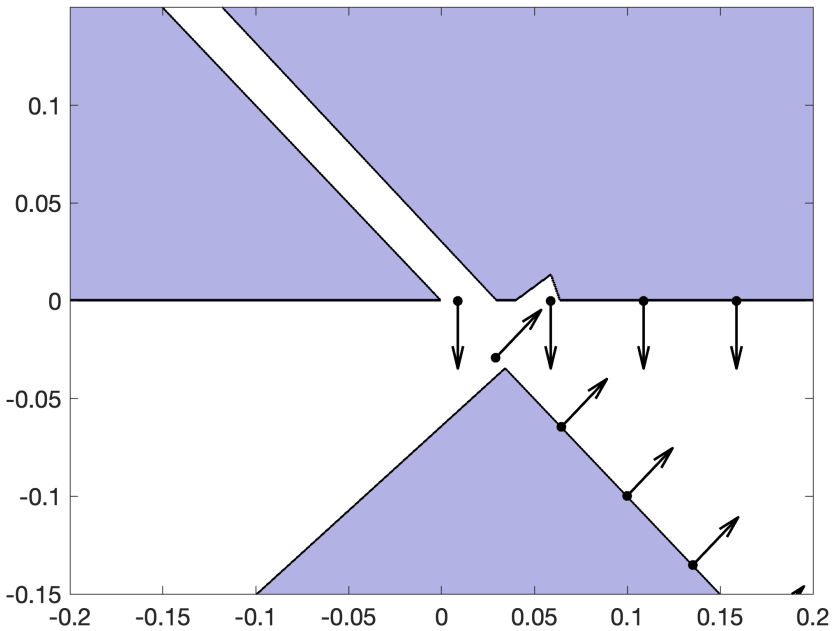
Easy fix: find closest boundary sample z^* first, then search K nearest boundary points around z^*. This gives a local set around z^*.

Boolean combinations of half-plane tests

Now for the serious issue. INNK-all combines Booleans from K half-space tests with logical conjunction (the AND operator). This is correct only for corners with an interior angle $\leq 180°$.



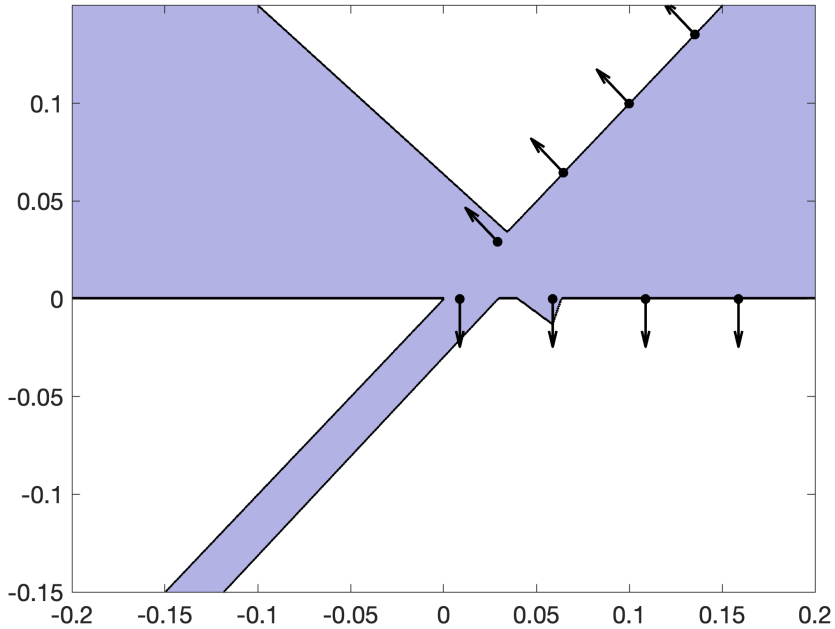
(a) 45° interior angle.



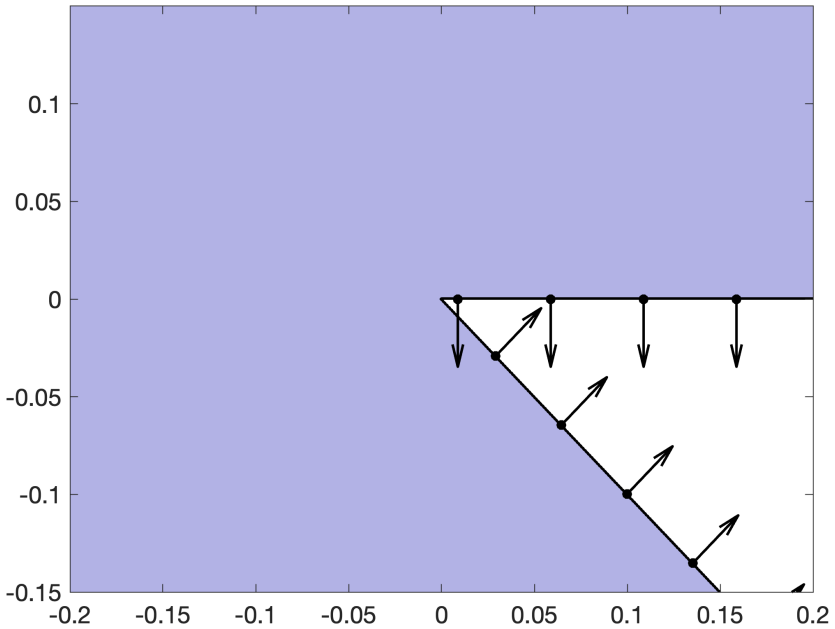
(b) 315° interior angle.

Boolean combinations of half-plane tests

Switching to a different test INNK-any that combines Booleans from K half-space tests with logical disjunction (the OR operator) does not fix the problem. More complex Boolean formulas also fail.



(a) 45° interior angle.



(b) 315° interior angle.

Boolean combinations of half-plane tests

Key observation

When the K half-space tests all return the same Boolean value, that value can be trusted. When the values are mixed, we say that the query point y and the local set of boundary samples around z^* are in an ambiguous configuration.

Query point relocation

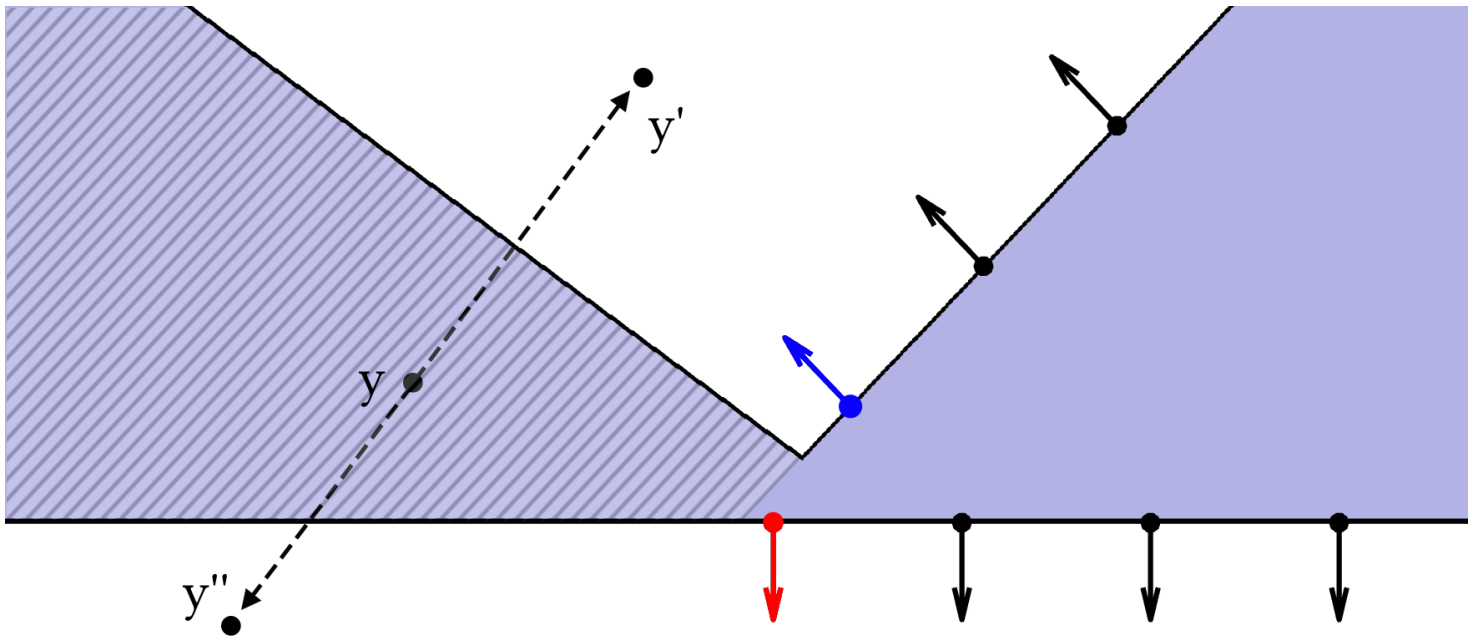
The distance $\|z_* - y\|$, if larger than $1.5h_Z$, approximates sufficiently well the distance of y from $\partial\Omega$. Let v be a unit vector orthogonal to $z_* - y$. Then,

$$y' = y + \alpha \|z_* - y\| v, \quad \alpha = 0.8$$

is a new query point such that $y' \in \Omega \iff y \in \Omega$ and, crucially, may not belong to an ambiguous configuration anymore.

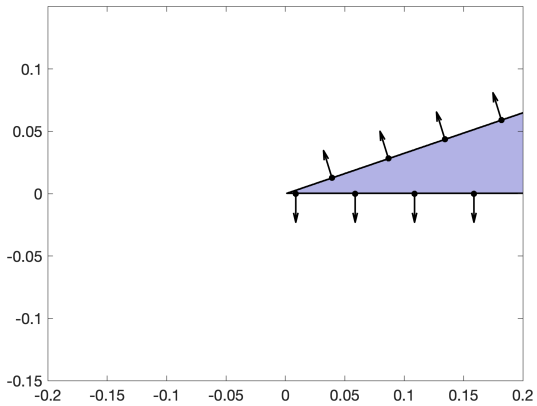
INNK-relocate

Each ambiguous query point leads to d recursive relocations in pseudo-random directions v locally parallel to the boundary.
In this way, we try to move away from nonsmooth features of $\partial\Omega$.
A maximum recursive depth is fixed (say, 3). If we are too close to the boundary or only ambiguous configuration are found, we return false.

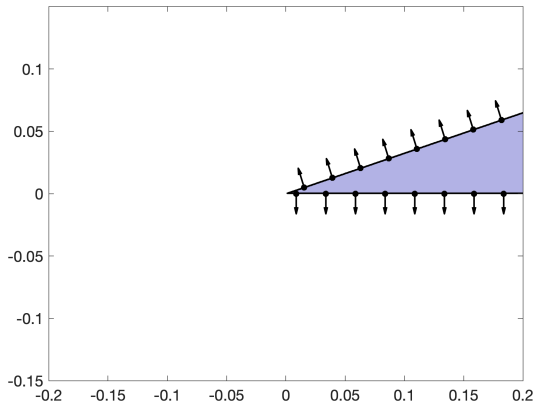


INNK-relocate

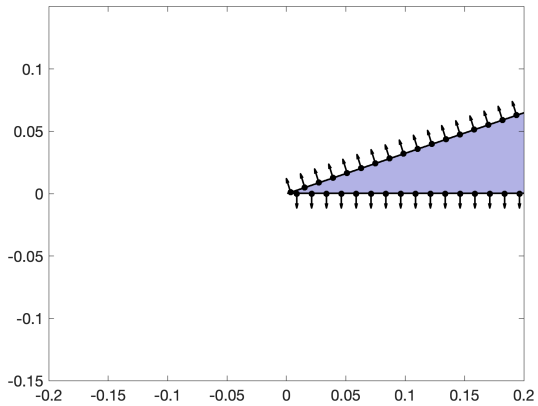
The inclusion test has empirical order of convergence 1 around corners.



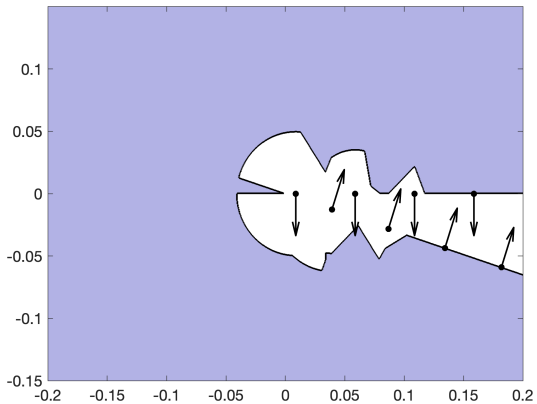
(a) $h_Z = 0.05$



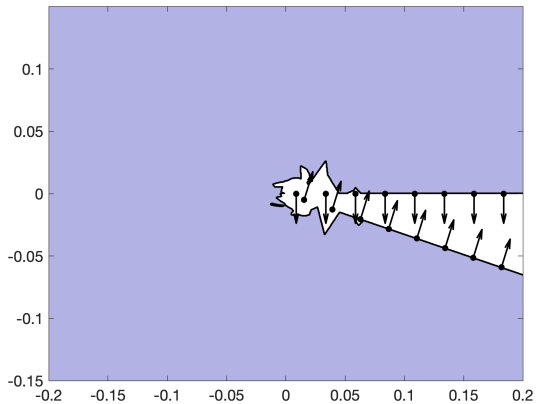
(b) $h_Z = 0.03$



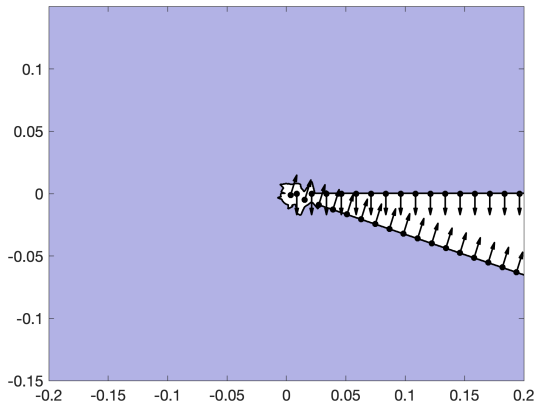
(c) $h_Z = 0.01$



(d) $h_Z = 0.05$



(e) $h_Z = 0.03$



(f) $h_Z = 0.01$

Neural inclusion tests

Neural inclusion tests

We have seen that a Boolean function of the signs of the scalar products

$$(z_i - y) \cdot \nu_i, \qquad i = 1, \ldots, K$$

is not a good way to combine local geometric information.

Local neural approach

We train a domain-independent neural network to learn a function

$$F(y, z_1, \nu_1, \ldots, z_K, \nu_K)$$

that we can use for point membership classification.

We seek a network as small as possible, to keep the cost of inference comparable to a k-d tree search. We avoid the expensive recursive searches of INNK-relocate and achieve higher accuracy by making better use of geometric information already available.

Construction of synthetic datasets

The success of the neural approach depends on the construction of synthetic datasets with features $(y, z_1, \nu_1, \ldots, z_K, \nu_K)$ coming from smooth and nonsmooth domains and their local sets of boundary points. Domains are chosen so that implicit representations are known and inclusion labels can be computed exactly.

2D dataset

Smooth features with different curvatures and nonsmooth features with different interior angles from 9° to 351°, for a total of 400k rows.

3D dataset

All sorts of tori, solid angles, cylinder sectors, pyramids, Fichera corners, and non-Lipschitz domains, for a total of 1M rows.

Architecture and training

The network is a multi-layer perceptron trained with Adam, binary cross-entropy loss and L2 regularization. Loss of each batch element is scaled by a function of the distance of y from $\partial\Omega$.

We measure classification accuracy, sensitivity, specificity, and maximum distance μ from $\partial\Omega$ of misclassified points (very important metric!).

We achieve the best outcome for a given network size by not using as inputs the raw features, but rather the scalar products

$$(z_i - y) \cdot \nu_i, \qquad i = 1, \ldots, K,$$

and the additional values

$$(z_i - z_j) \cdot \nu_i, \qquad i, j = 1, \ldots, K, \quad i \neq j.$$

Architecture and training

We have optimized hyperparameters by a grid search with optuna, and verified stability with respect to random weights initialization and random split of dataset (70% training, 20% validation, 10% test).

2D hyperparameters

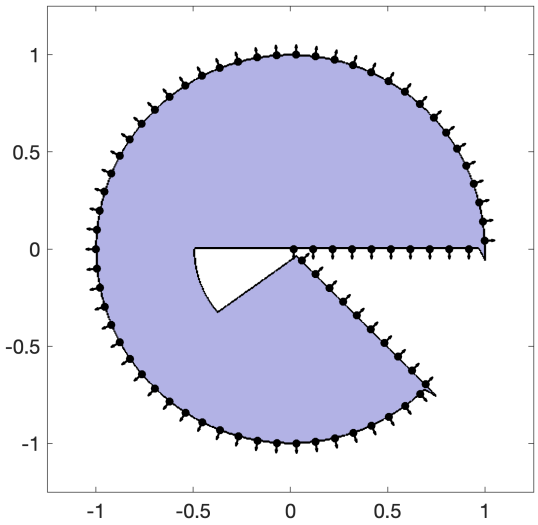
$K = 2$, 3 layers, hidden size 8, learning rate 0.001 for 2000 epochs, weight decay 0, and batch size 1024. Overall size: 193 parameters.

3D hyperparameters

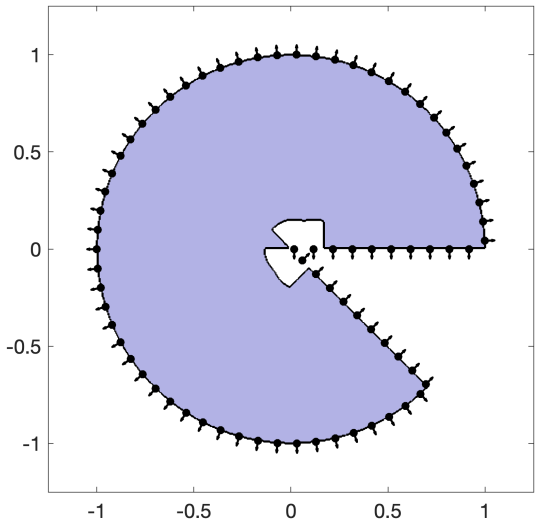
$K = 4$, 4 layers, hidden size 32, learning rate 0.001 for 1000 epochs, weight decay 0.01, and batch size 1024. Overall size: 5281 parameters.

Results in the 2D case

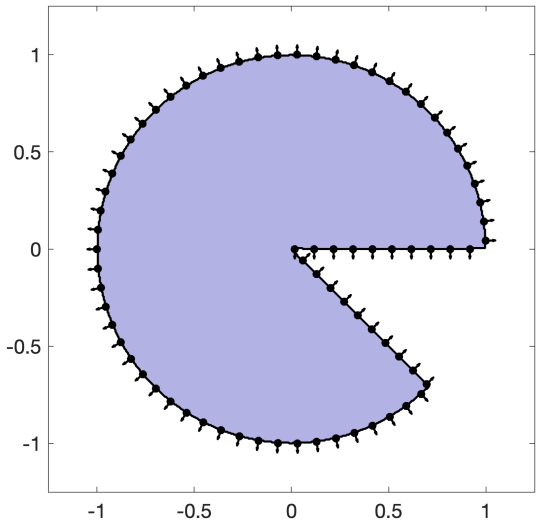
The neural approach captures 2D corners almost perfectly: compare the reconstructed domains for 315° internal angle.



(a) INN1



(b) INN2-relocate



(c) INN2-neural

Results in the 2D and 3D case

Performance on the synthetic 2D dataset,
with oversampling of ambiguous configurations:

- INN1: accuracy 0.7336, $\mu = 30h_Z$
- INN2-relocate: accuracy 0.9809, $\mu = 1.5h_Z$
- INN2-neural: accuracy 0.9991, $\mu = 0.81h_Z$

Performance on the synthetic 3D dataset,
without oversampling of ambiguous configurations:

- INN1: accuracy 0.9852, $\mu = 73h_Z$
- INN8-relocate: accuracy 0.9804, $\mu = 1.5h_Z$
- INN4-neural: accuracy 0.9987, $\mu = 0.88h_Z$

Thank you for your attention!