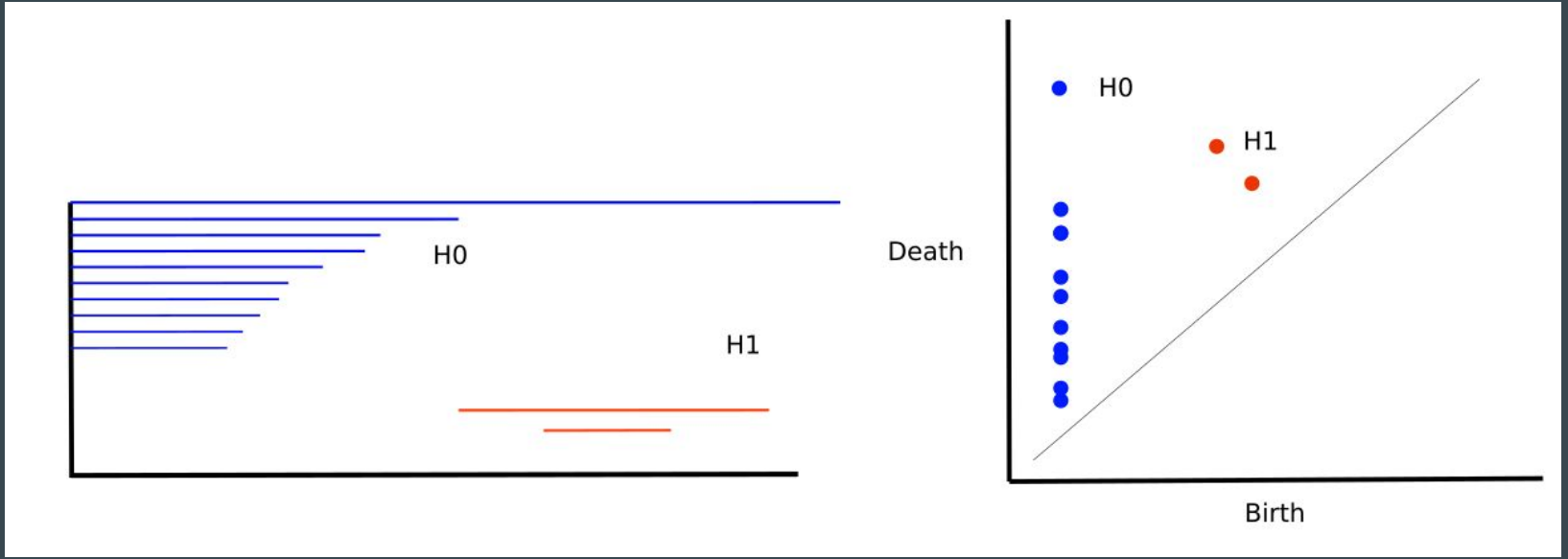
# Algebraic topology of neurons

...

# Persistence diagrams

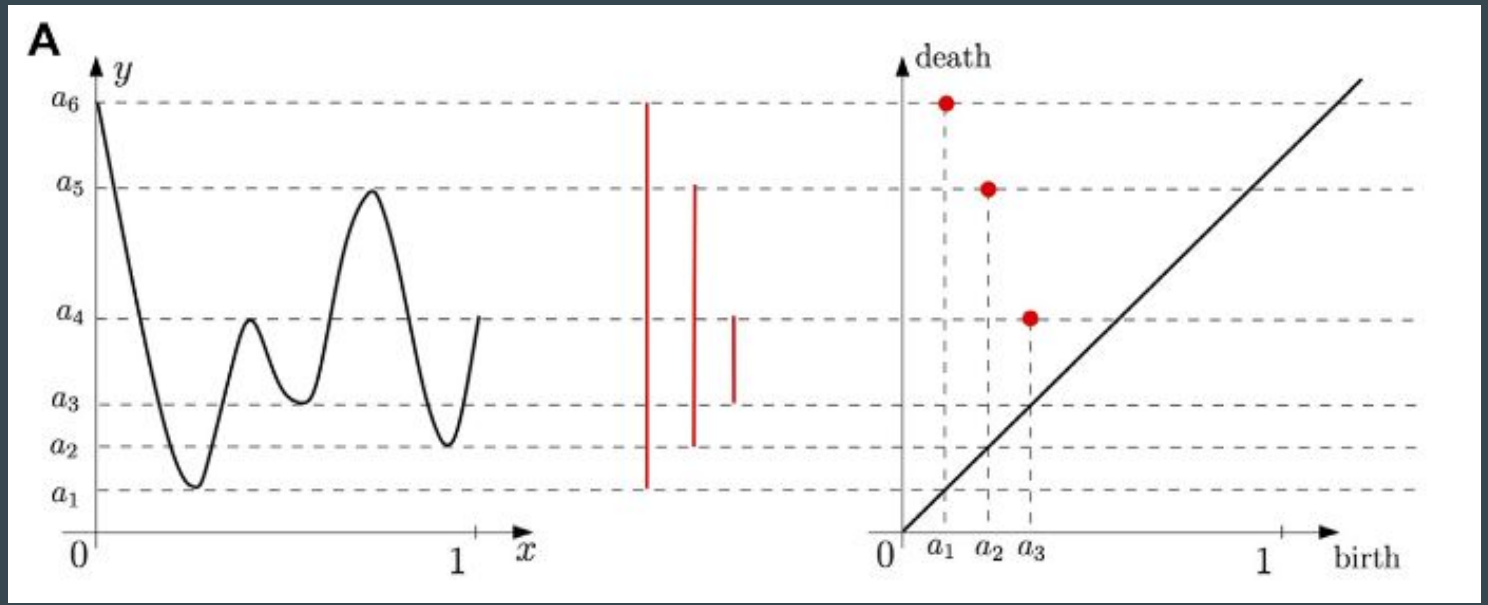
# Filtration over a point cloud: Vietoris-Rips or Cech

Persistence diagrams (barcodes) represent the evolution of topological features with respect to a parameter: here size of spheres epsilon



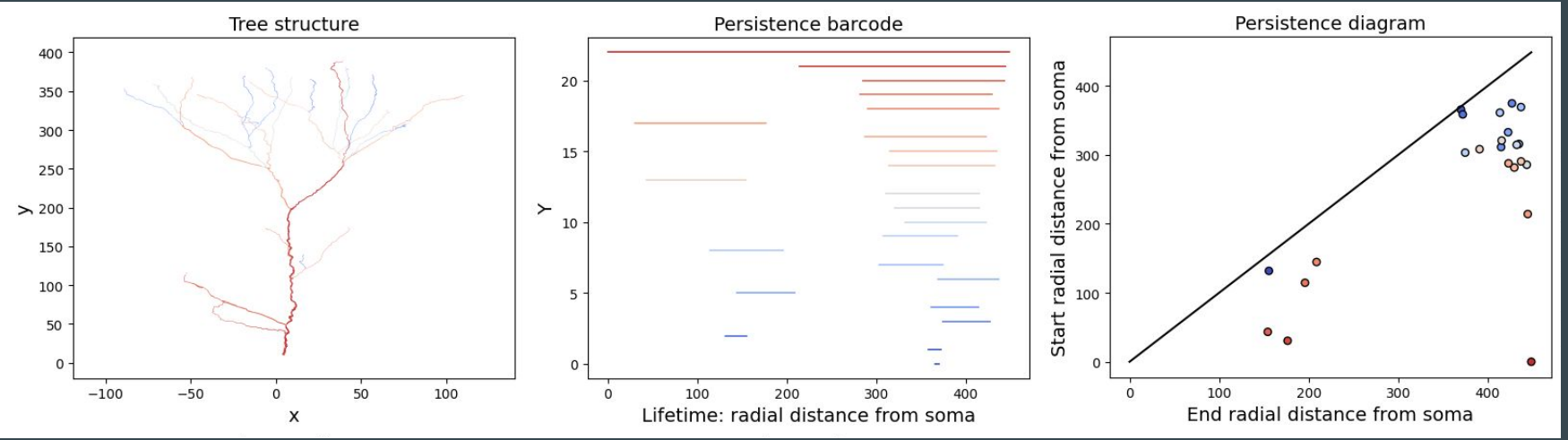
# Filtration over a point cloud: Height filtration

Persistence diagrams (barcodes) represent the evolution of topological features with respect to a parameter: here height of a curve



# Filtration of a tree: Topological Morphology Descriptor

In TMD we compute the evolution of topological features by traversing the tree with respect to a function f (radial distance, path distance etc...)



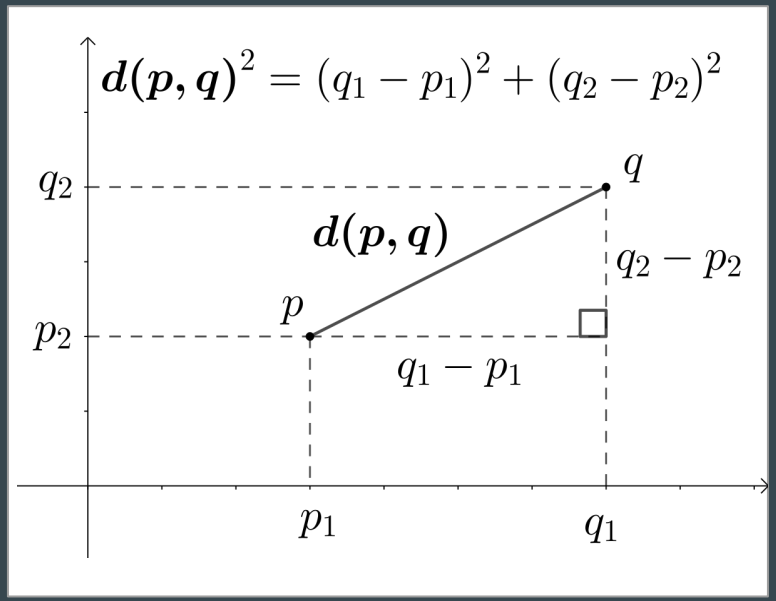
# Once we extract the persistence diagrams...

How can we compare them, compute distances and statistics between them and eventually use them with machine learning tools?

# Persistence distances

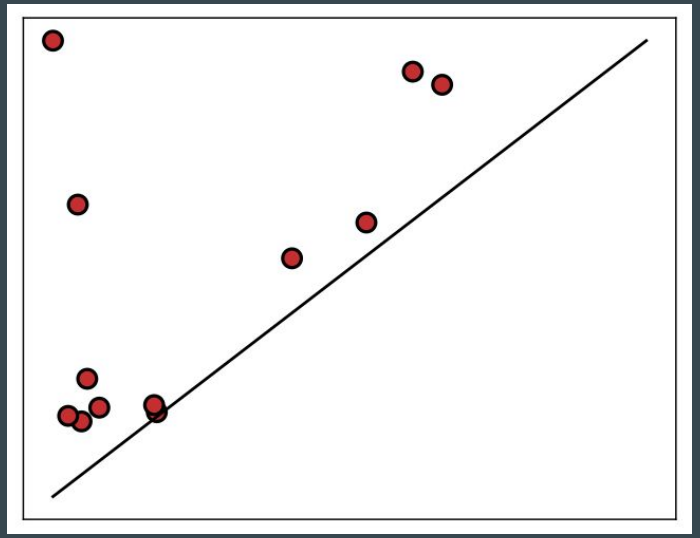
# How do we compute distances between diagrams?

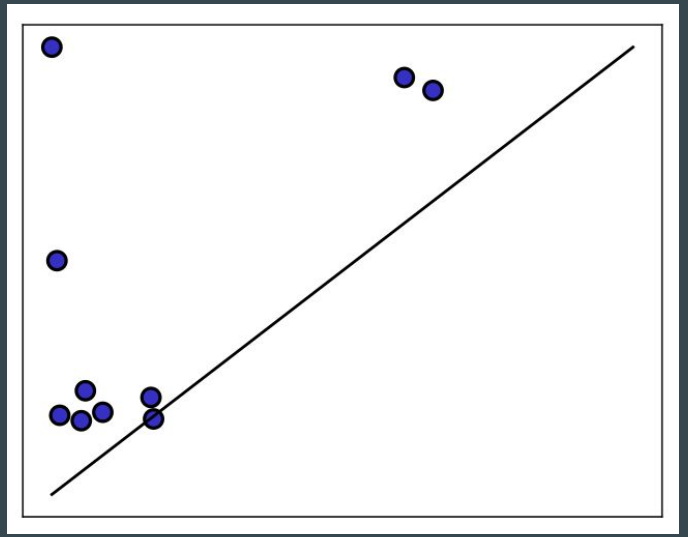
In Euclidean space is it “easy” to compute distances between points by computing the difference in each dimension:



# How do we compute distances between diagrams?

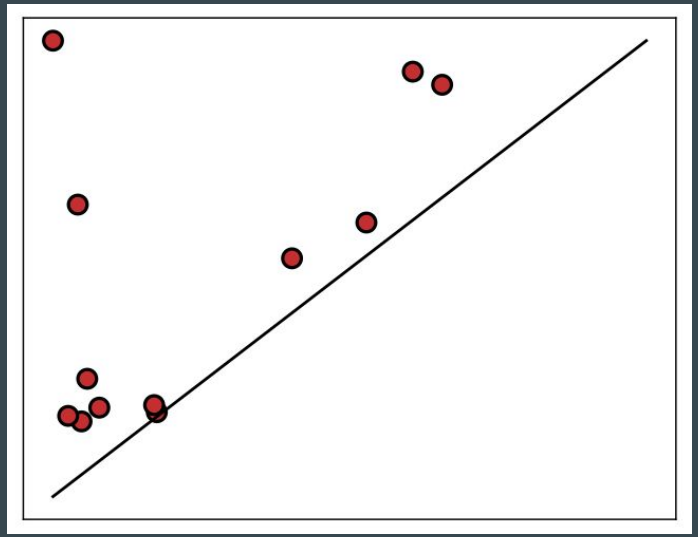
In the space of persistence diagrams we have a few problems to compute distances...

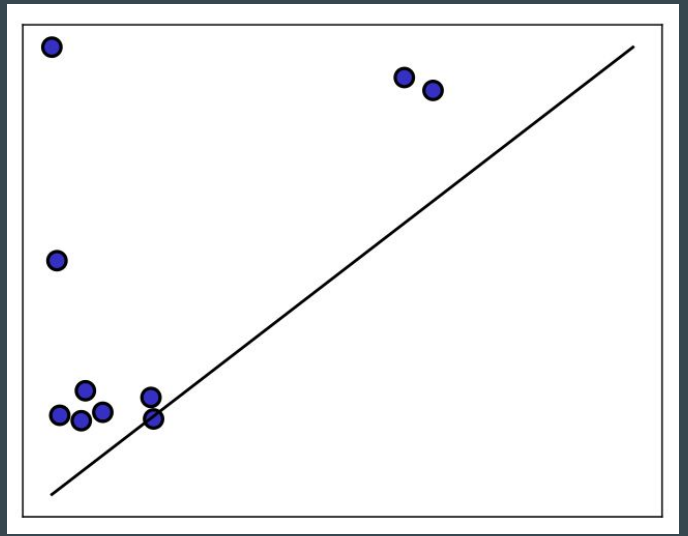




# How do we compute distances between diagrams?

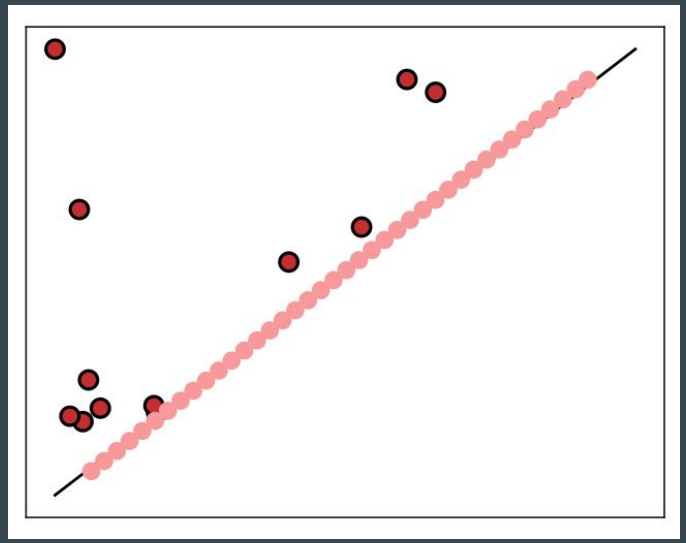
Problem #1 in persistence diagrams: how to match dimensions? For example, red diagram has 12 points while blue has 10 points

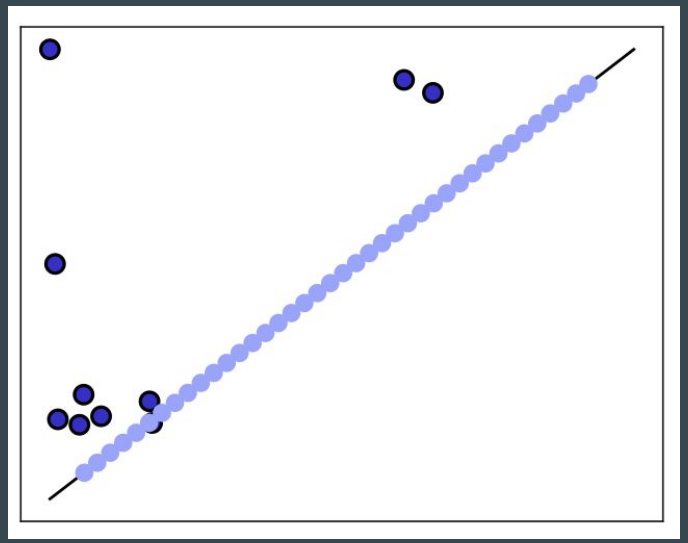




# How do we compute distances between diagrams?

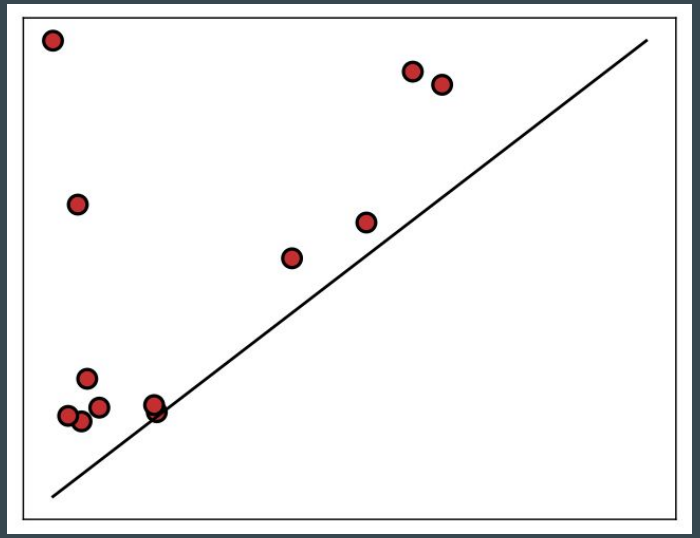
Problem #1 in persistence diagrams: how to match dimensions? By adding infinitely many points on the diagonal of the diagrams.

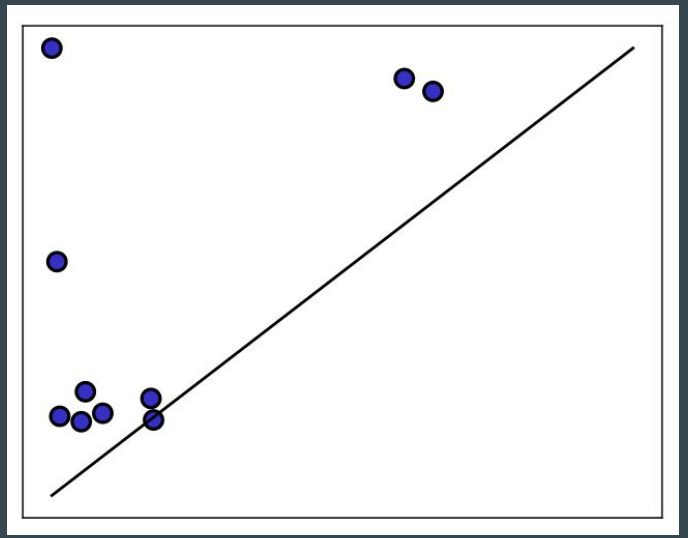




# How do we compute distances between diagrams?

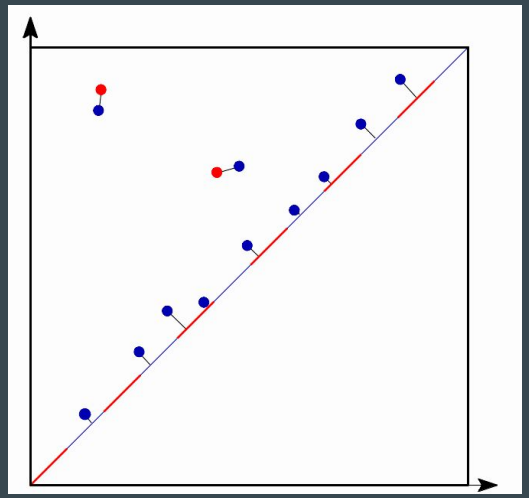
Problem #2 in persistence diagrams: how to align dimensions? Which points in the red diagram should be matched to each point in the blue diagram?





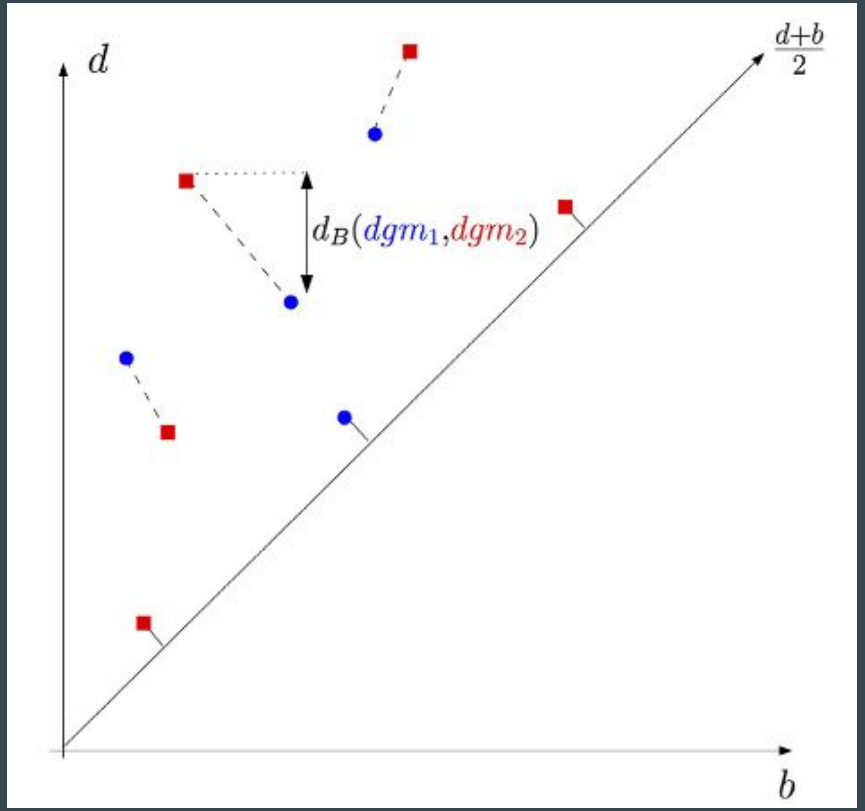
# How do we compute distances between diagrams?

Problem #2 in persistence diagrams: how to align dimensions? Optimization algorithm so that some quantity is minimized: red points matched either to blue points or points in the diagonal. The optimal matching depends on the distance!



# Bottleneck distance

The bottleneck distance measures the similarity between two persistence diagrams. It is the shortest distance dB for which there exists a perfect matching between the points of the two diagrams (completed with all the points on the diagonal in order to ignore cardinality mismatchs) such that any couple of matched points are at distance at most dB




Chazal and Michel 2021

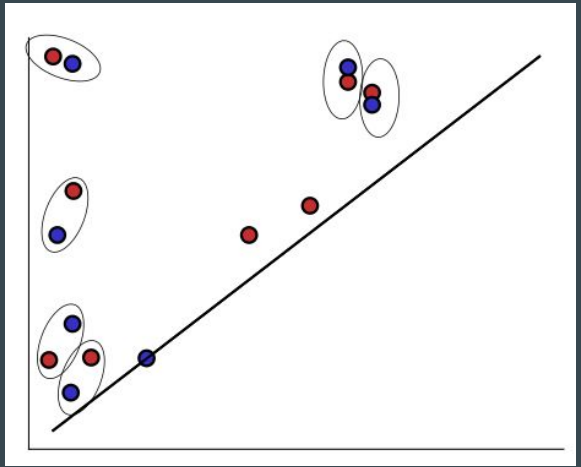
# Bottleneck distance

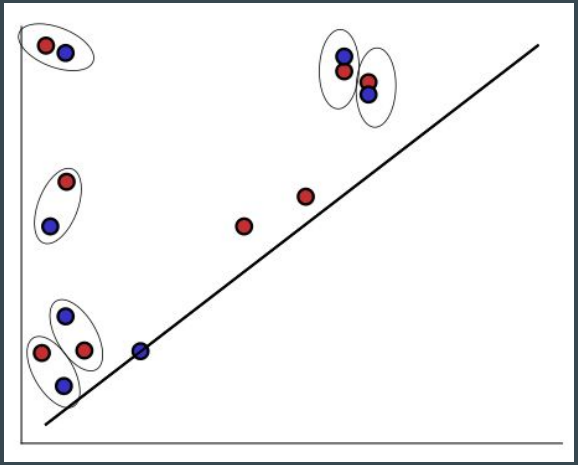
Let X and Y be two persistence diagrams. To define the distance between them, we consider bijections $\eta : X \to Y$ and record the supremum of the distances between corresponding points for each. Measuring distance between points $x = (x1 , x2 )$ and $y = (y1 , y2 )$ as $||x - y||_\infty = \max\{|x\,1 - y\,1\,|, |x\,2 - y\,2\,|\}$ and taking the infimum over all bijections, we get the bottleneck distance between the diagrams:

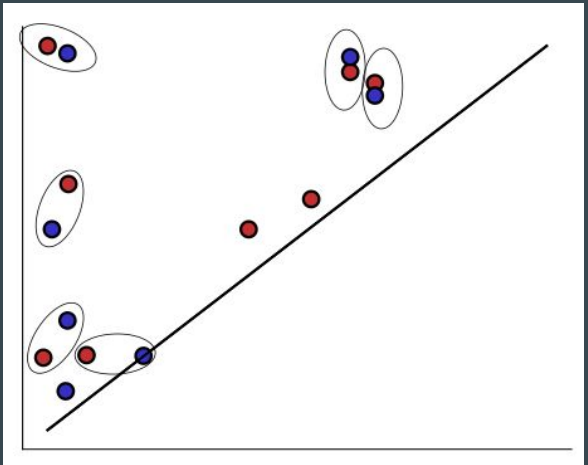
$$W_\infty(X,Y) \quad = \quad \inf_{\eta: X \to Y} \sup_{x \in X} \|x - \eta(x)\|_\infty$$

# Bottleneck distance

Intuitively, the dB(X, Y) is the maximum cost of the most efficient of all possible matchings between X and Y diagrams.







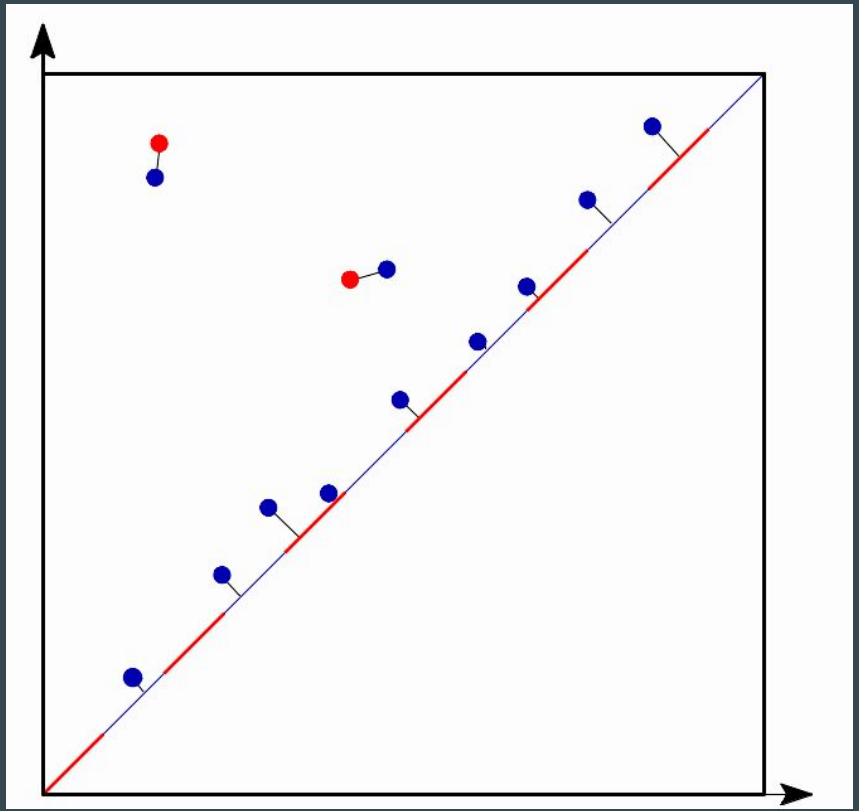
# Wasserstein distance

A drawback of the bottleneck distance is its insensitivity to details of the bijection beyond the furthest pair of corresponding points. To remedy this shortcoming, we introduce the degree q Wasserstein distance between X and Y for any positive real number q. It takes the sum of q-th powers of the $L_\infty$-distances between corresponding points, again minimizing over all bijections

$$W_q(X,Y) = \left[ \inf_{\eta: X \to Y} \sum_{x \in X} \|x - \eta(x)\|_\infty^q \right]^{1/q}$$

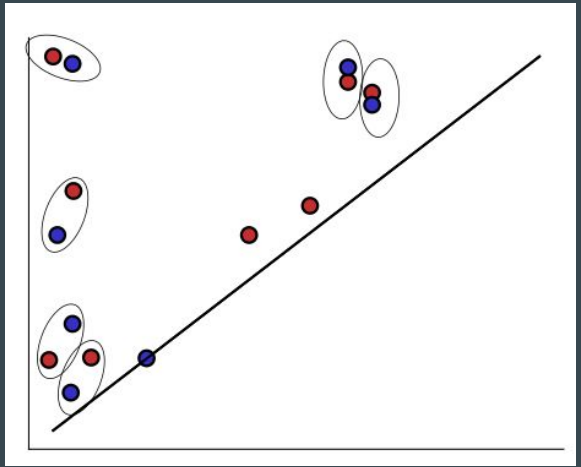
# Wasserstein distance

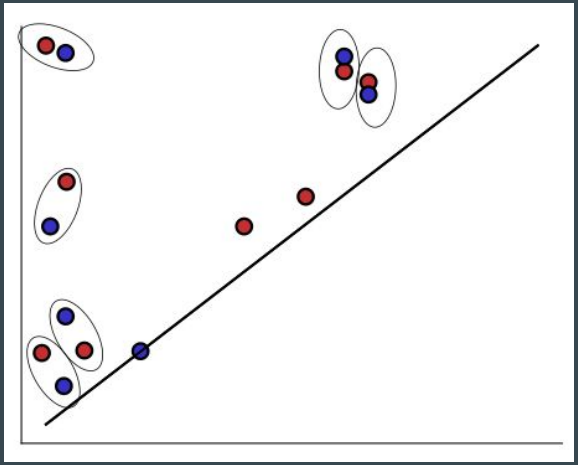
The Wasserstein distance measures the similarity between two persistence diagrams. It is the shortest distance dW for which there exists a perfect matching between the points of the two diagrams (completed with all the points on the diagonal in order to ignore cardinality mismatchs) such that the total p-distance between matched points is dW.

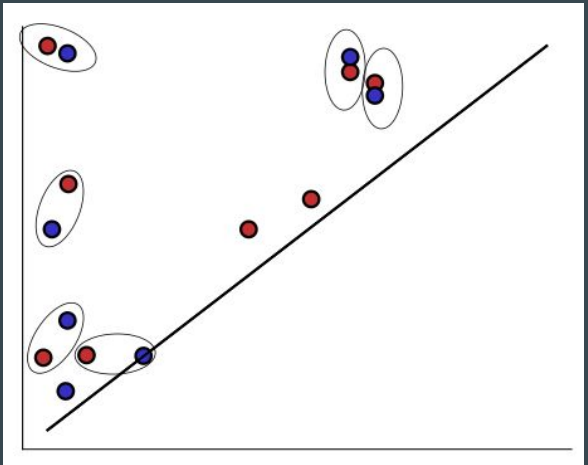


# Wasserstein distance

Similarly to the dB, the dW(X, Y) is the total cost of the most efficient of all possible matchings between X and Y diagrams.

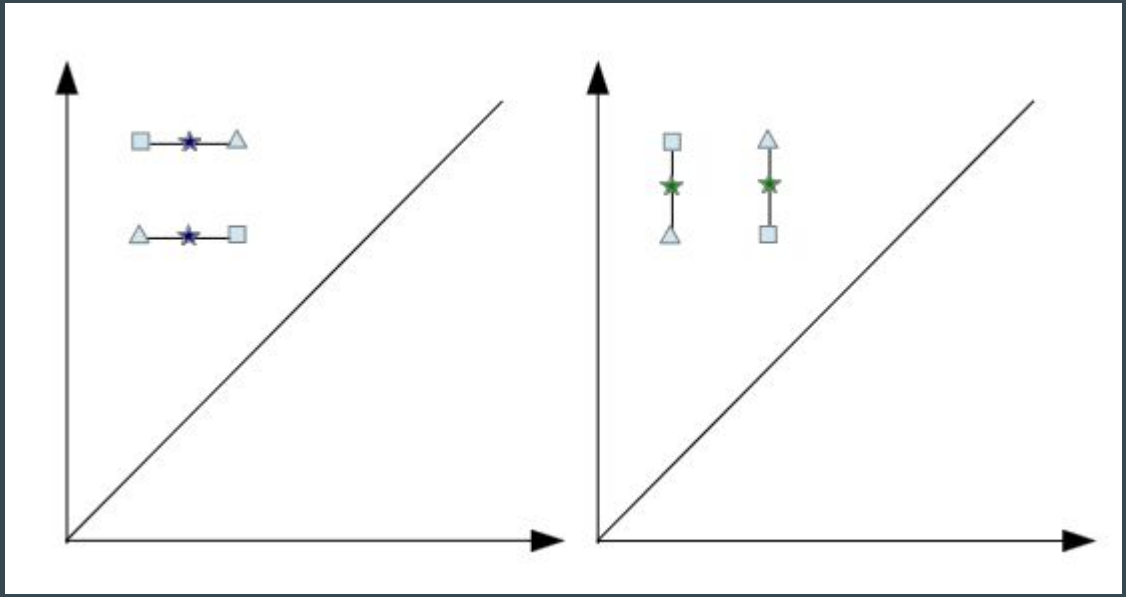






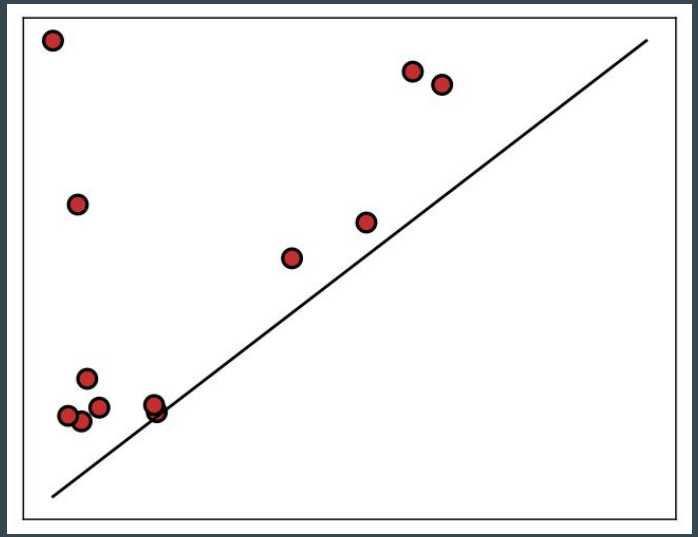
# How do we compute Frechet mean between diagrams?

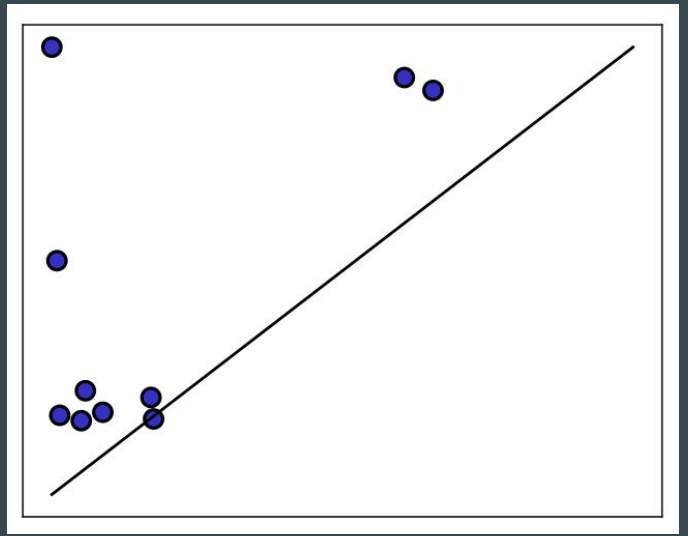
Problem #3: not clear definition of averages on persistence diagrams



# How do we compute distances between diagrams?

Another way to compute distances between two diagrams is by transforming them to another persistence representation





# Representations of persistence

# Representations of persistence

We can use different representations of persistence to define distances between diagrams that are not easy to define in the space of persistence barcodes or persistence diagrams:

Betti curves

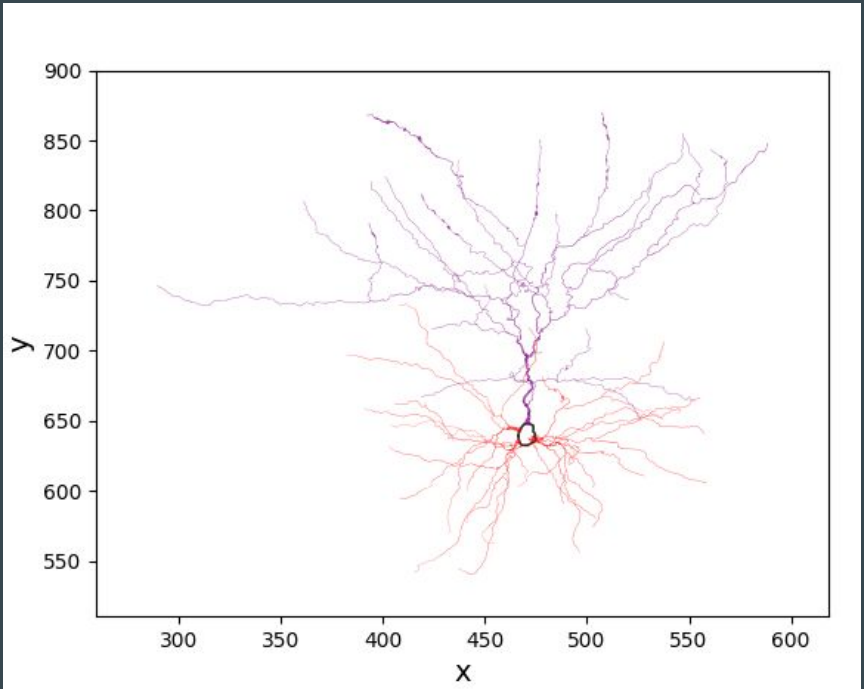
Persistence images

Landscapes

Simple statistics

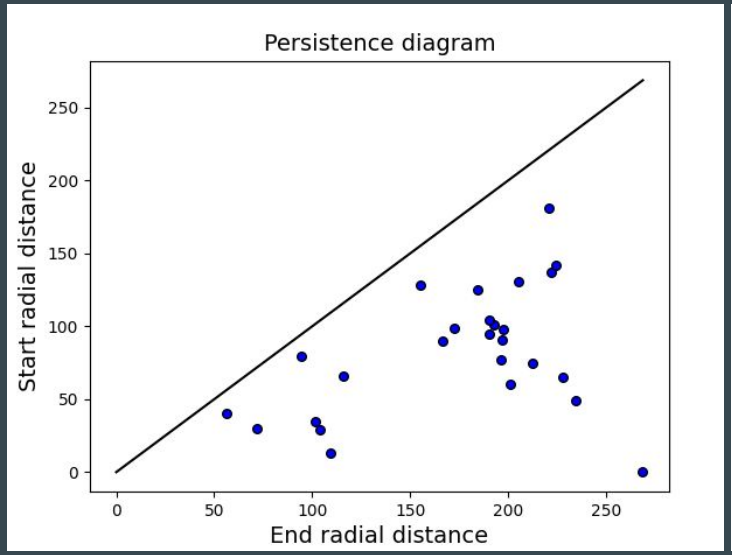
# Persistence of a neuronal morphology

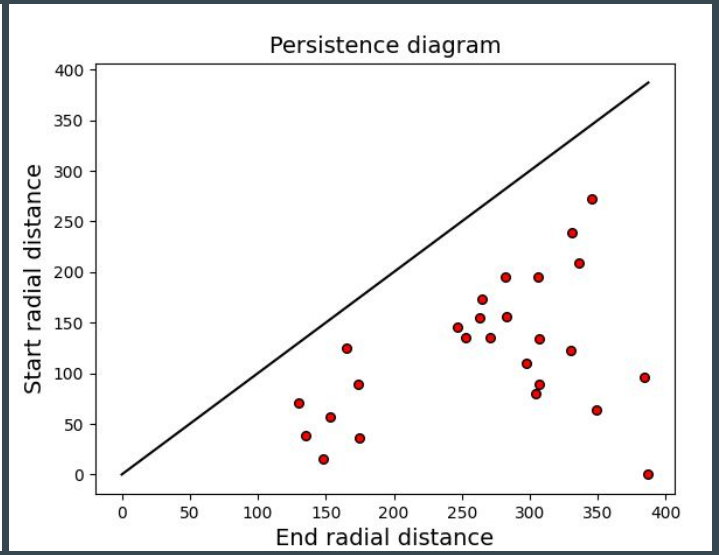
From the neuronal morphology we can extract persistence diagrams and barcodes



# Persistence of a neuronal morphology

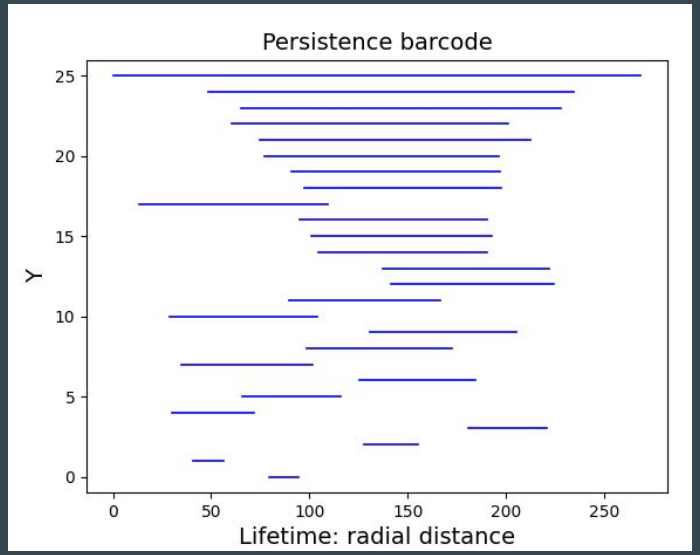
Computing the difference between radial and path distance diagrams:

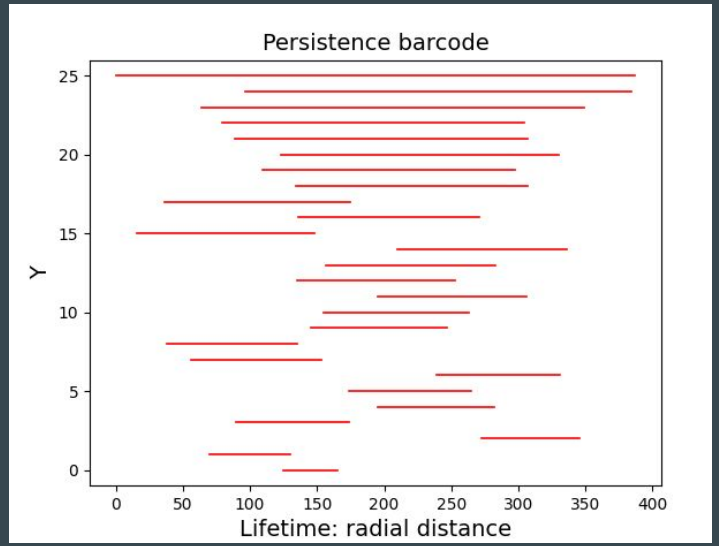


# Betti curves difference

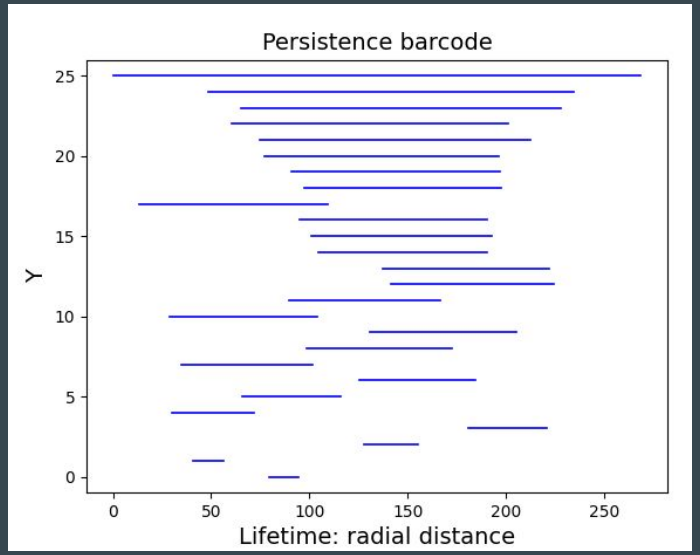
We can compute the difference between the Betti curves of the barcodes:

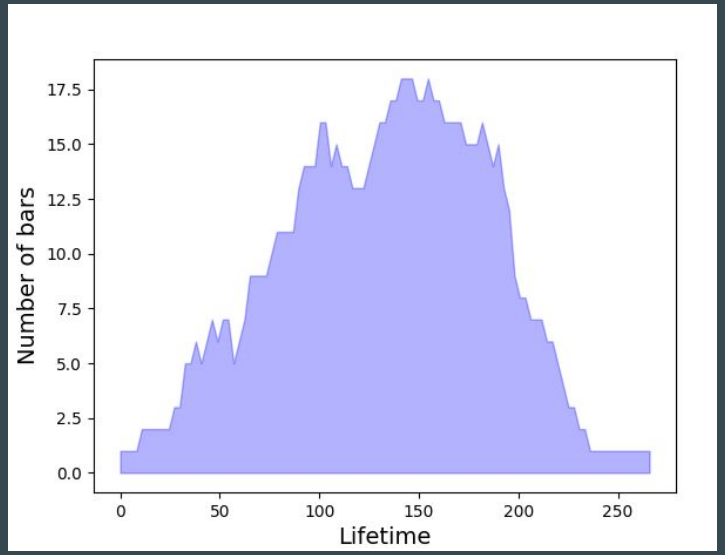


# Betti curves difference

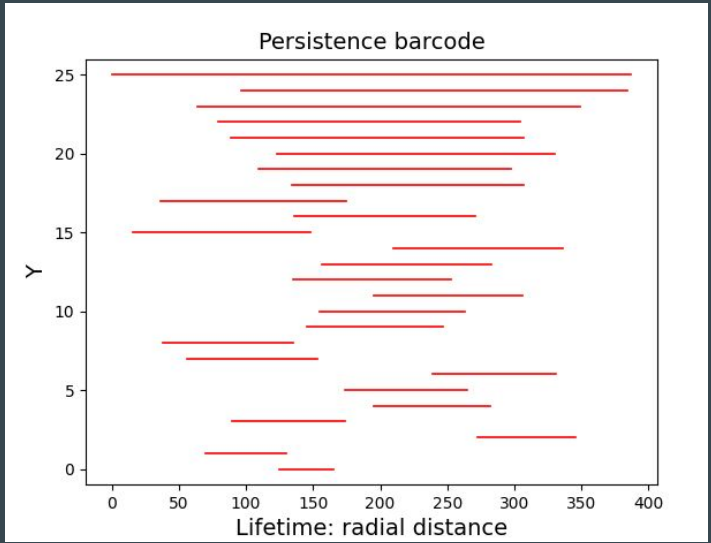
We can compute the difference between the Betti curves of the barcodes:

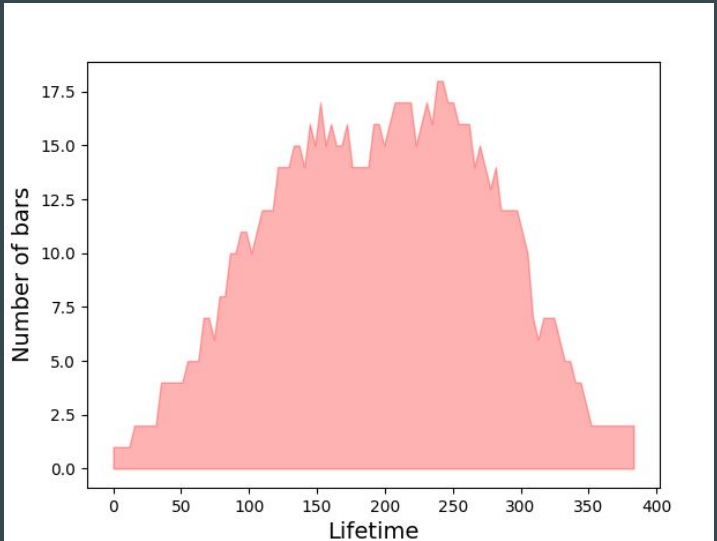


# Betti curves difference

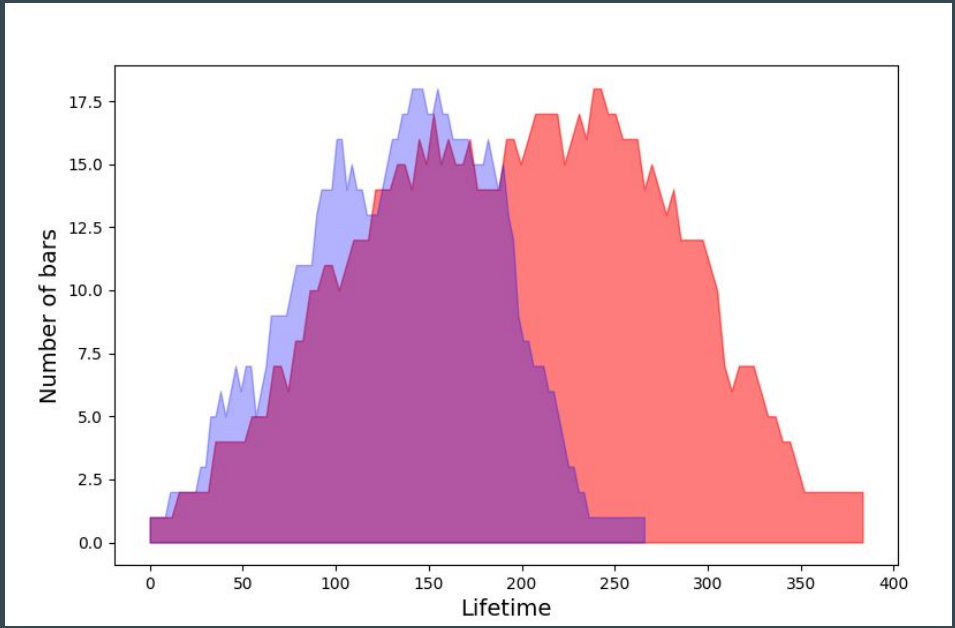
We can compute the difference between the Betti curves of the barcodes:



# Betti curves difference

In this case, the difference between the two diagrams can be computed by:



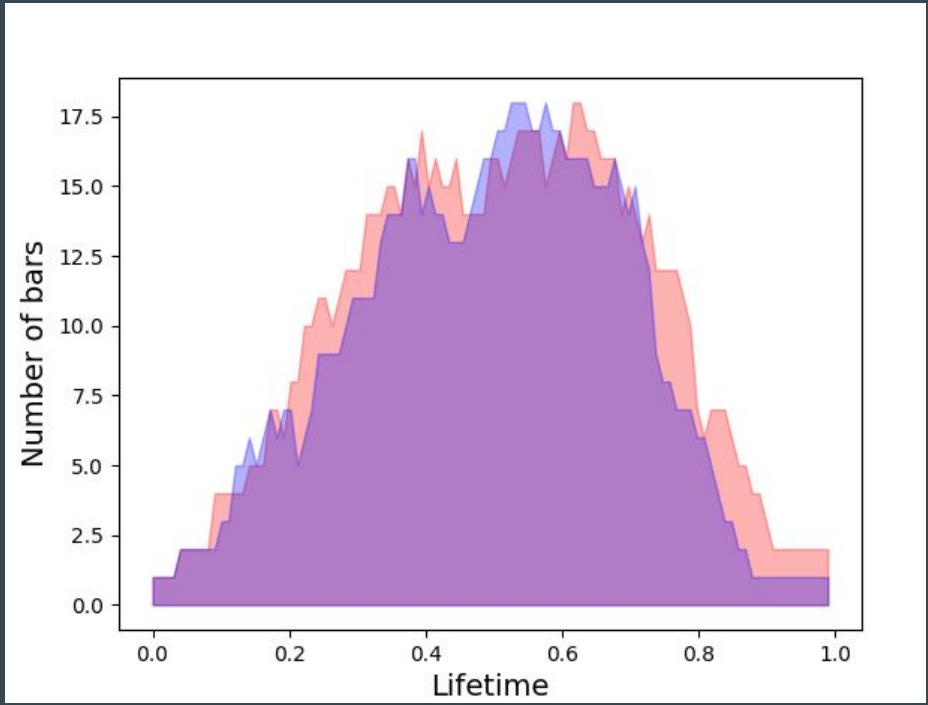

- total difference

- maximum difference

...

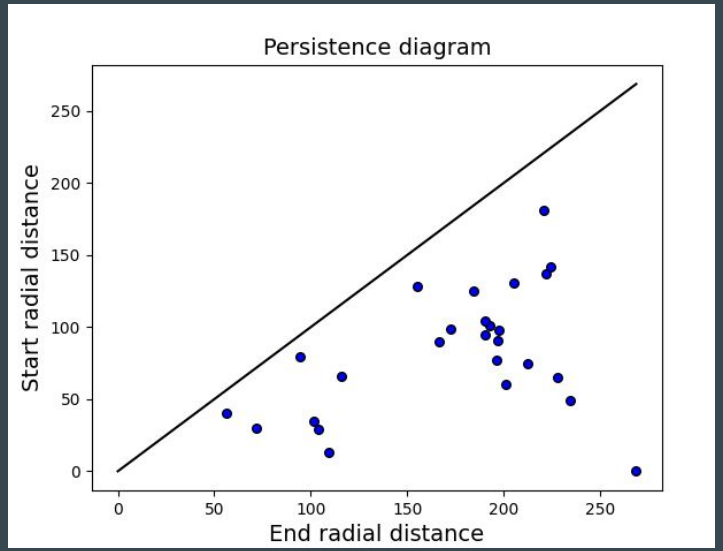
# Betti curves difference

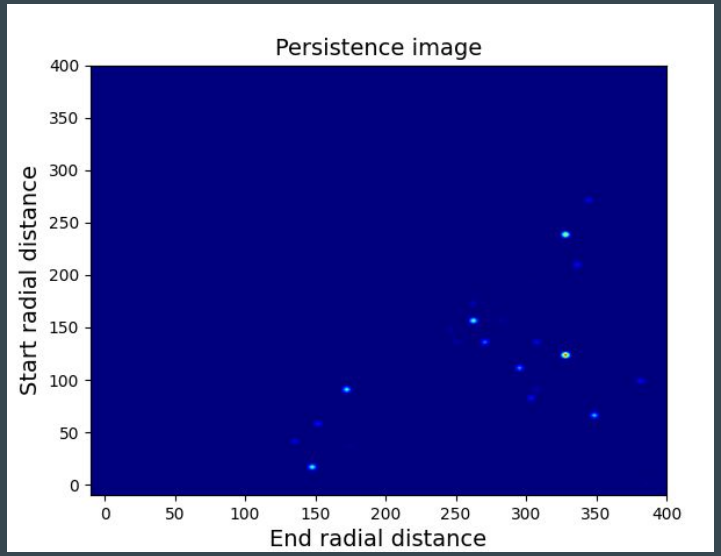
This difference will heavily depend on the normalization of the input data!



# Persistence images difference

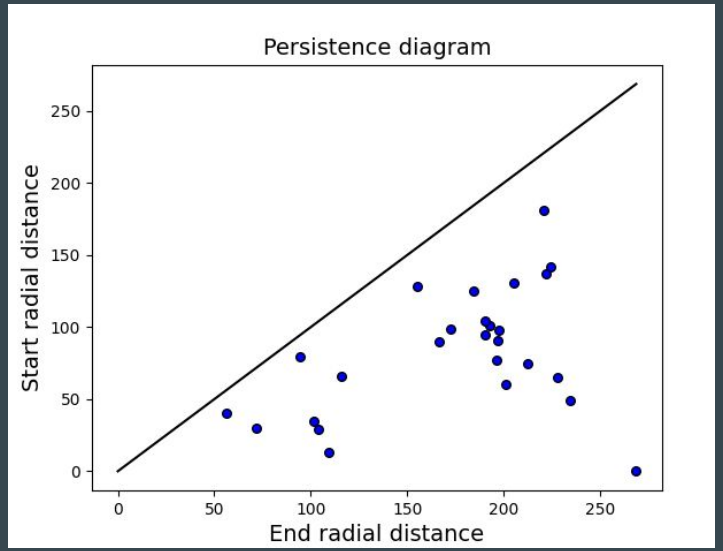
Persistence images are generated by gaussian kernel averages around the persistence points of the persistence diagram

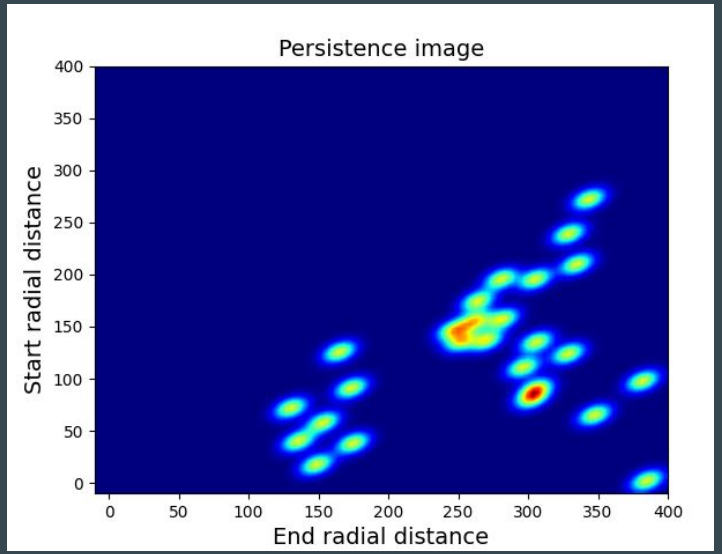


# Persistence images difference

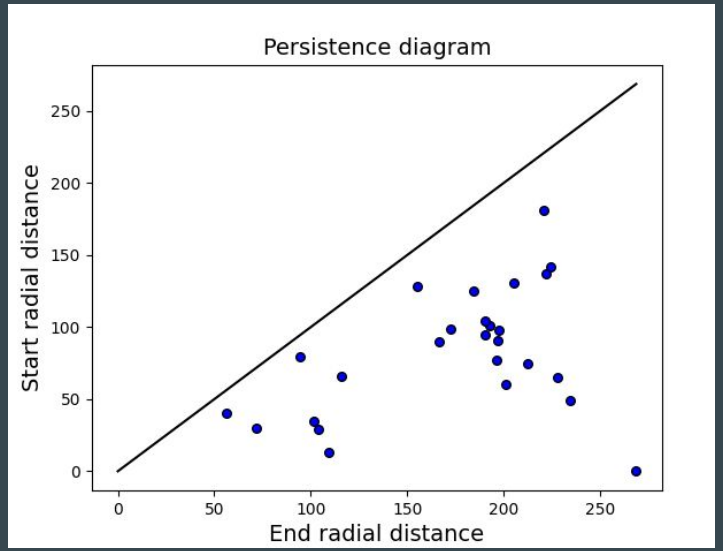
Persistence images are generated by gaussian kernel averages around the persistence points of the persistence diagram

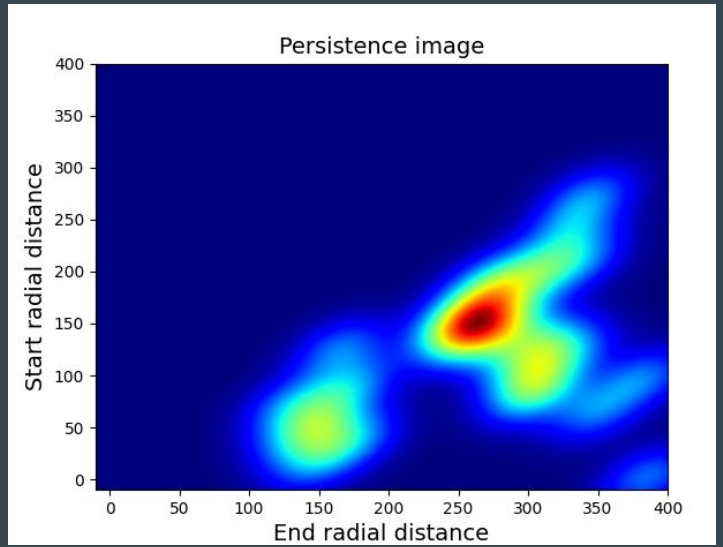


# Persistence images difference

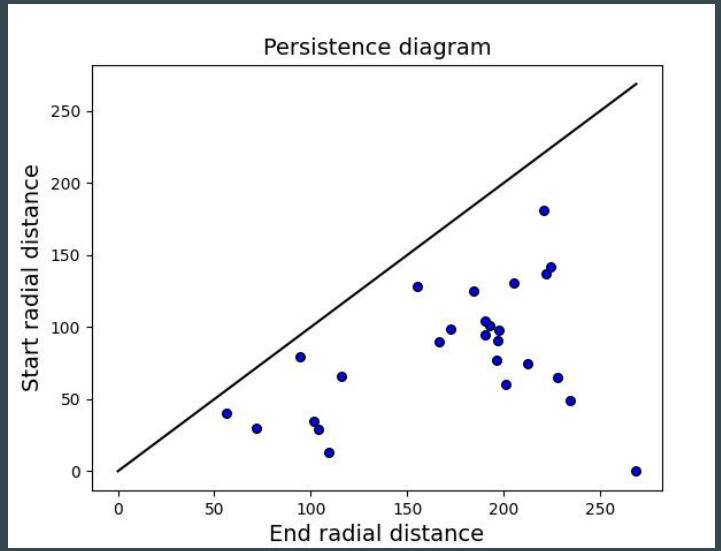
Persistence images are generated by gaussian kernel averages around the persistence points of the persistence diagram

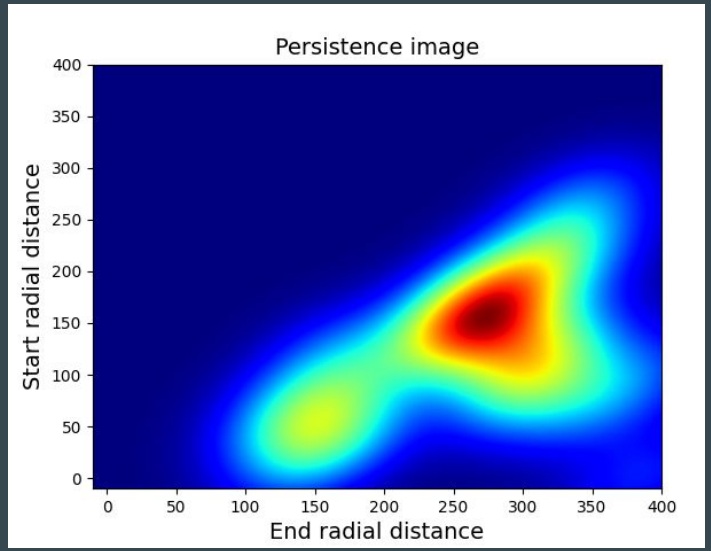


# Persistence images difference

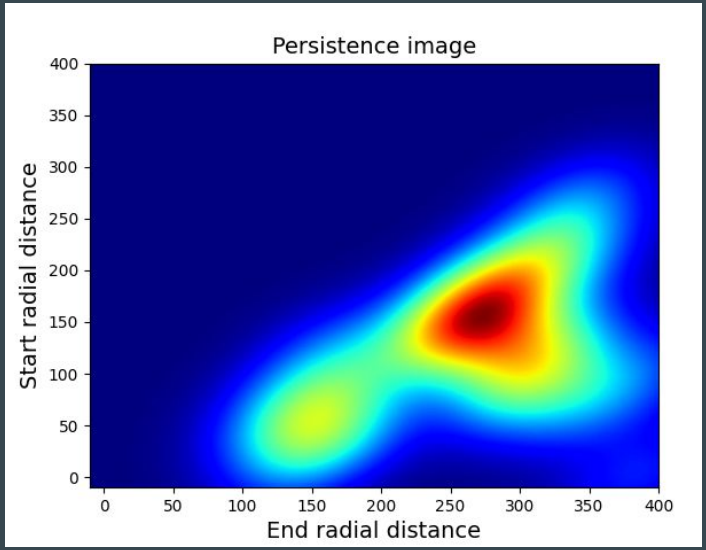
Persistence images are generated by gaussian kernel averages around the persistence points of the persistence diagram

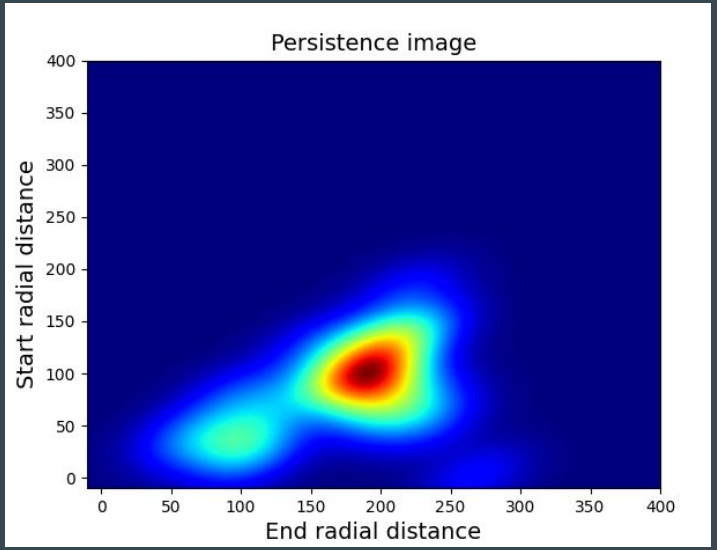


# Persistence images difference

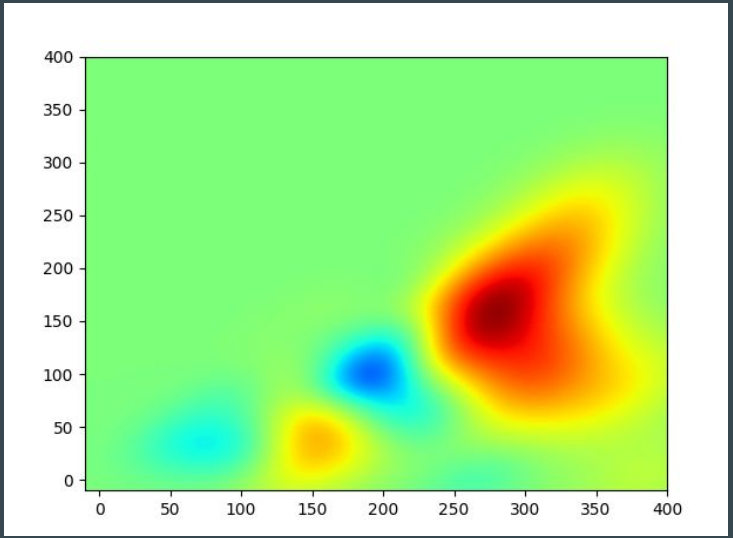
Persistence image difference can be computed based on the difference between each pixel value between two images



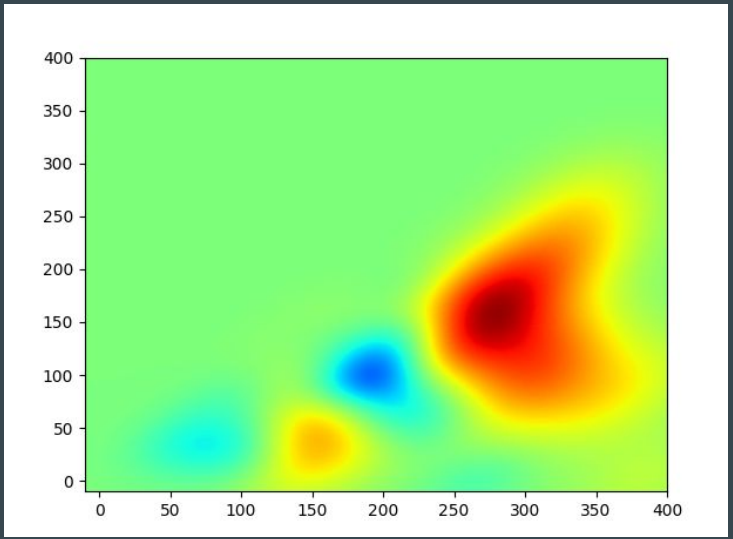
# Persistence images difference

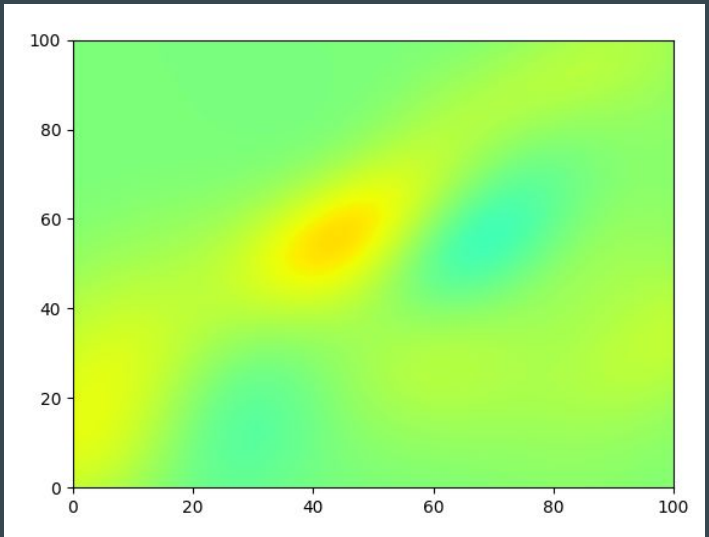
Persistence image difference can be computed based on the difference between each pixel value between two images



# Persistence images difference

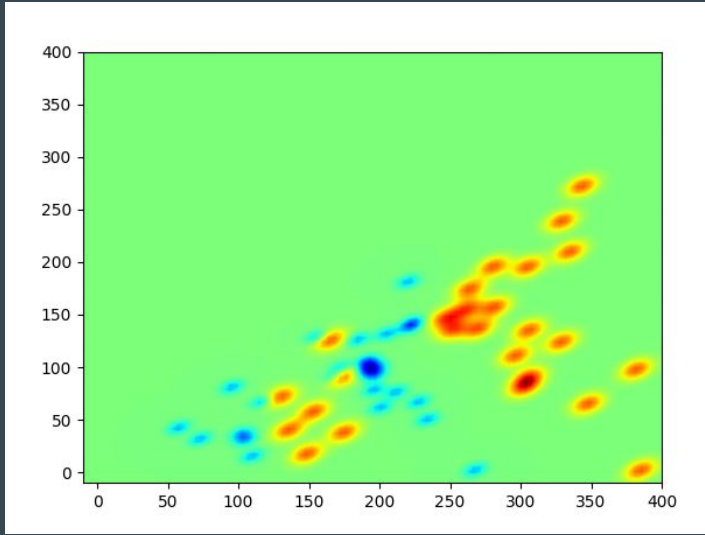
Normalization and parameters are also important for the computation of differences between persistence images:

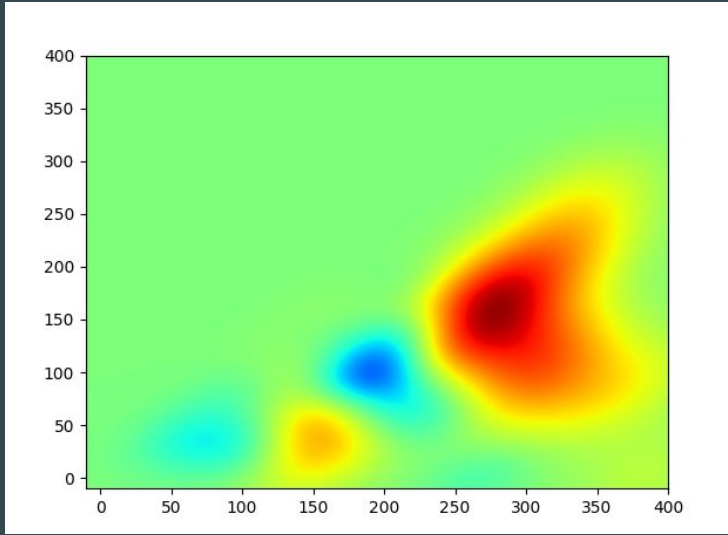


# Persistence images difference

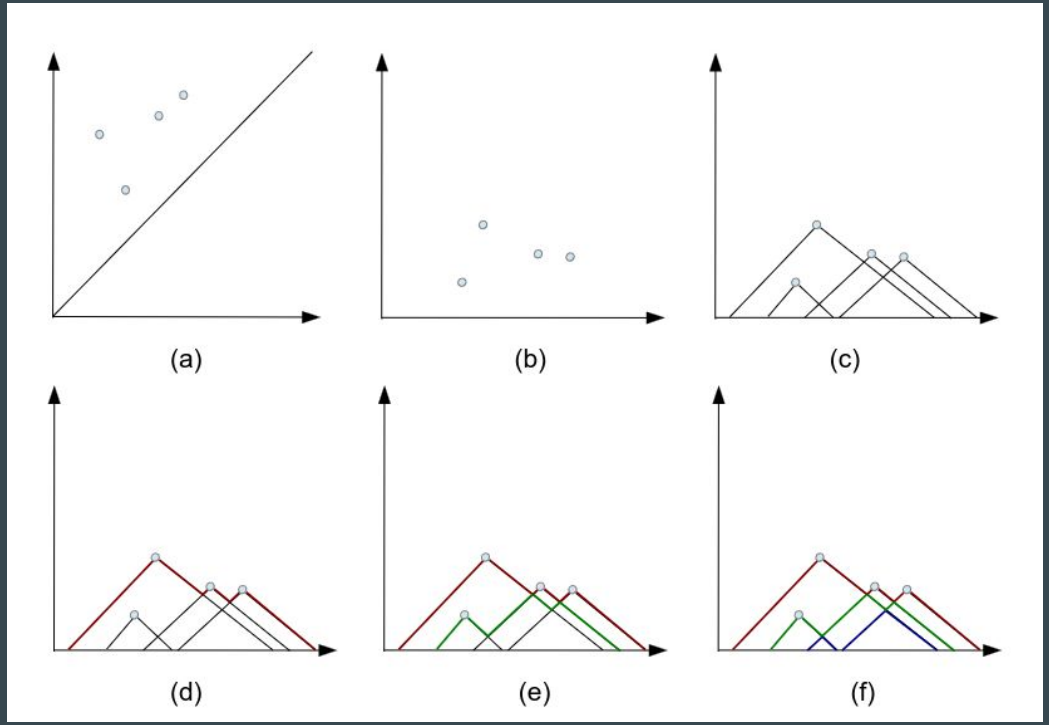
Normalization and parameters are also important for the computation of differences between persistence images:



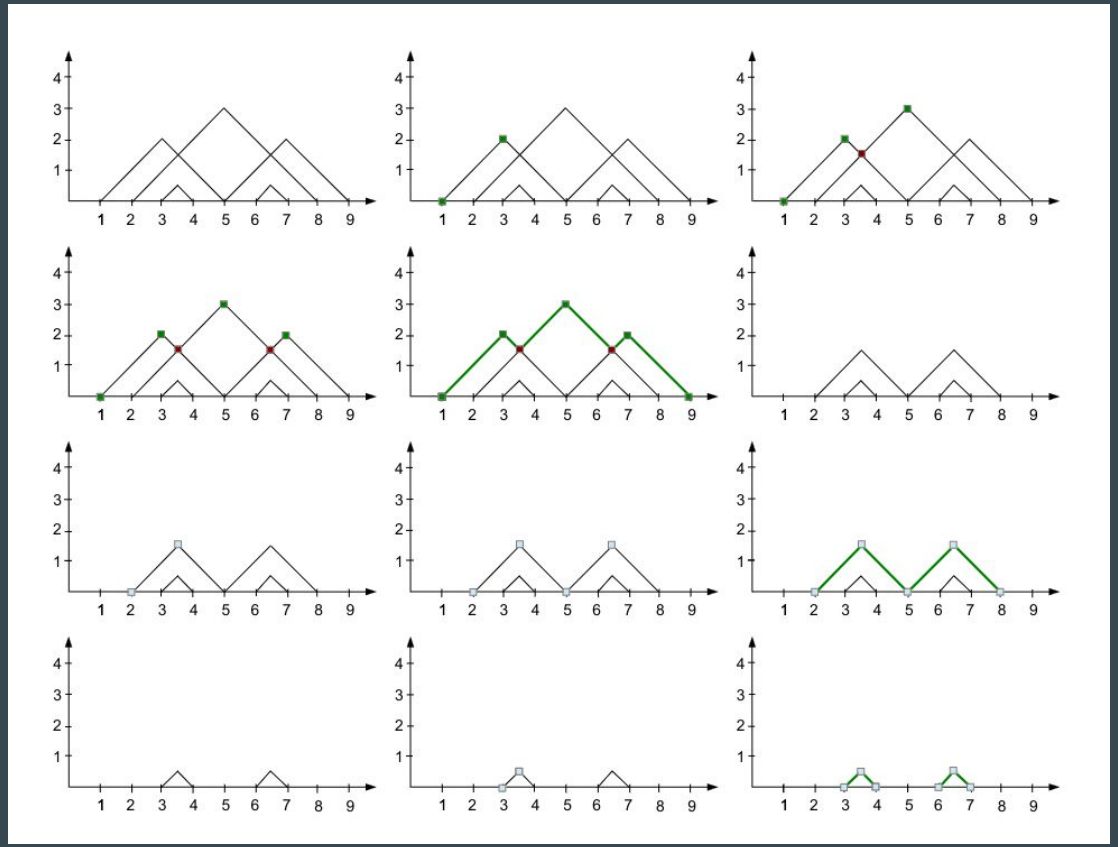
# Persistence Landscape

Persistence landscape. First, we ”lie down” the persistence diagram (b). Formally this corresponds to moving from (birth, death) coordinates into (middlife, half life) ones. Then, for the image of every interval (b, d) in the new coordinates, we draw a plot of the function f(b,d) as in (c). Then, the first landscape function, $\lambda 1$ is depicted in Figure (d), the $\lambda 2$ in (e) and $\lambda 3$ in (f)



# Persistence Landscape

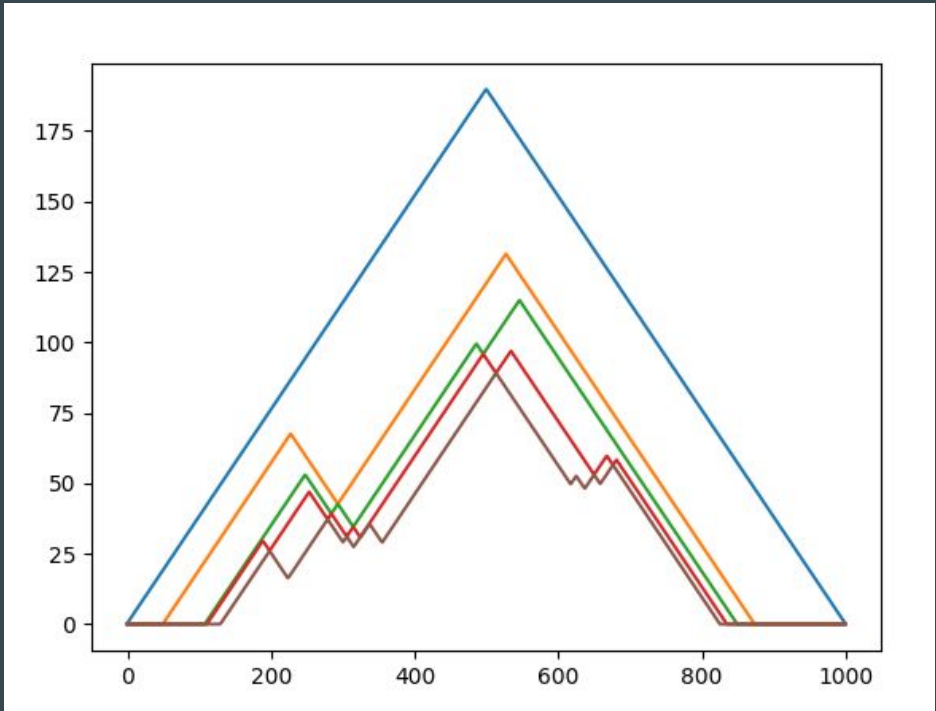
Algorithm used to construct the persistence landscape corresponding to the birth-death pairs {(1, 5), (2, 8), (3, 4), (5, 9), (6, 7)}. The first three critical points define λ1 (max green line). The remaining pairs (red points) generate the second landscape λ2 (second green line) until the final λ3 is generated by the remaining two peaks.



# Landscape difference

Similarly to the previous representations, a vector can also be extracted from the persistence landscapes.

For example for the neuronal morphology, the persistence landscape is:

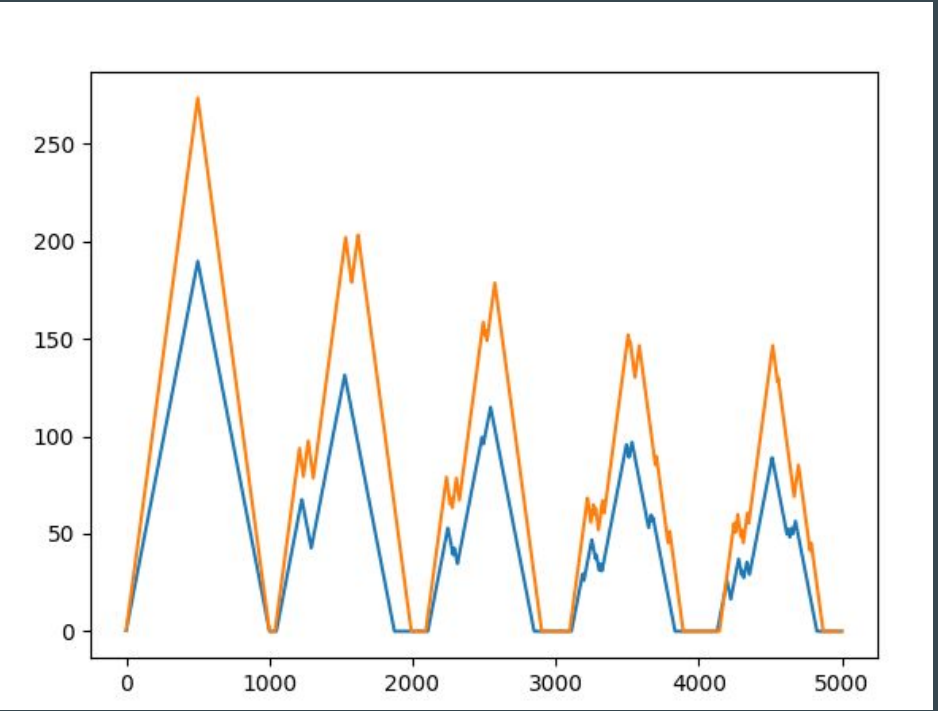


# Landscape difference

The vector created by adding all landscapes together can be used for the computation of the difference between landscapes by extracting:

- Max difference
- Total difference

...



# Persistence diagrams statistics

Another way to vectorize persistence is to extract a list of statistical properties from the diagram. These can be for example:

Persistence Statistics [dim = 0]

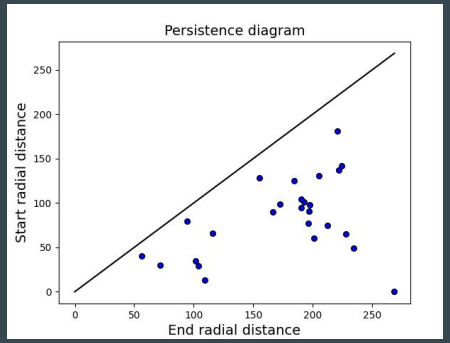
| | Mean | STD | Median | IQR | Range | P10 | P25 | P75 | P90 |
|---|---|---|---|---|---|---|---|---|---|
| Births | 94.2704 | 30.8133 | 96.0000 | 35.0000 | 130.0000 | 52.0000 | 78.0000 | 113.0000 | 138.0000 |
| Deaths | 109.0687 | 24.6201 | 110.0000 | 26.0000 | 216.0000 | 80.0000 | 96.0000 | 122.0000 | 139.0000 |
| Midpoints | 101.6695 | 24.4832 | 99.0000 | 24.5000 | 112.0000 | 68.0000 | 87.0000 | 120.0000 | 139.0000 |
| Lifespans | 14.7983 | 26.7117 | 5.0000 | 10.0000 | 238.0000 | 1.0000 | 2.0000 | 12.0000 | 48.0000 |

https://persistent-homology.streamlit.app/

# Classification based on persistence

# Classification based on persistence

The basic idea for the use of persistence for classification tasks, is the need to vectorize the persistence:



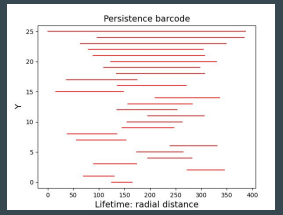

$[x_1, x_2, x_3, ..., x_n]$

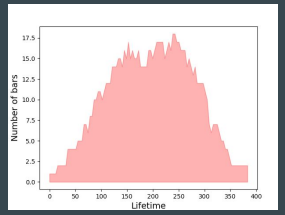
$[y_1, y_2, y_3, ..., y_n]$

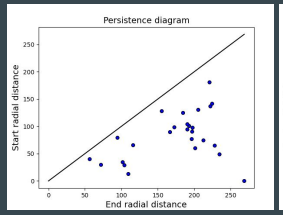
$[k_1, k_2, k_3, ..., k_n]$

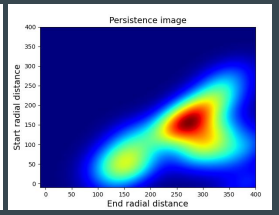
# Classification based on persistence

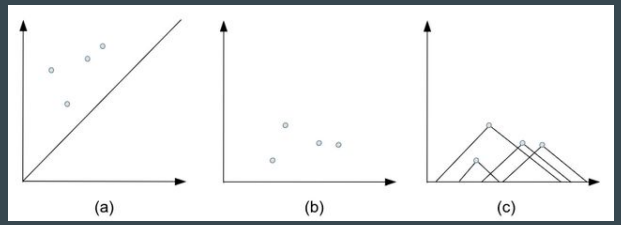
The basic idea for the use of persistence for classification tasks, is the need to vectorize the persistence, as described in the previous examples:

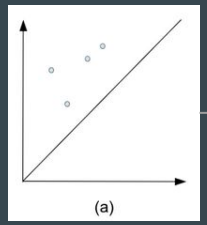



Statistics:
# points,
Mean birth
Mean death ...

# Classification based on persistence

For more ideas about vectorization techniques you can take a look at:

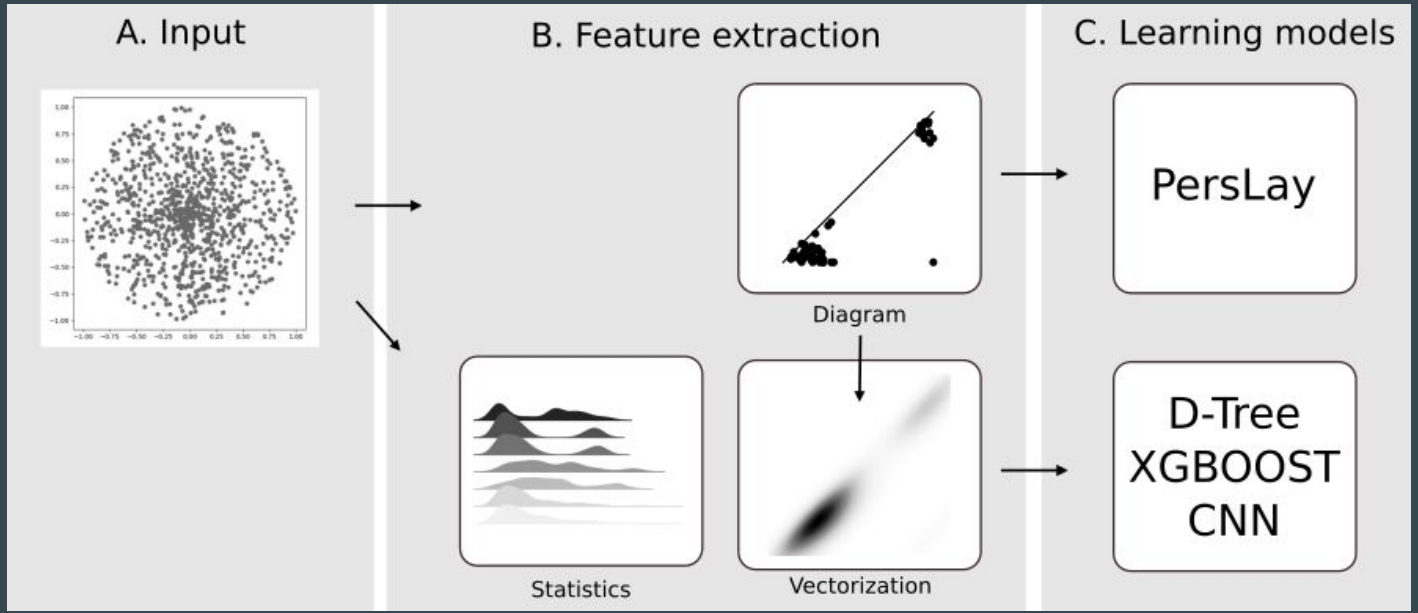
A Survey of Vectorization Methods in Topological Data Analysis, Ali et al. 2022

And try out their web app:

https://persistent-homology.streamlit.app/

# Classification based on persistence

The basic idea for the use of persistence for classification tasks, is the need to vectorize the persistence, then any ML algorithm can be used as in other datasets.



# A Topological Representation of Branching Neuronal Morphologies

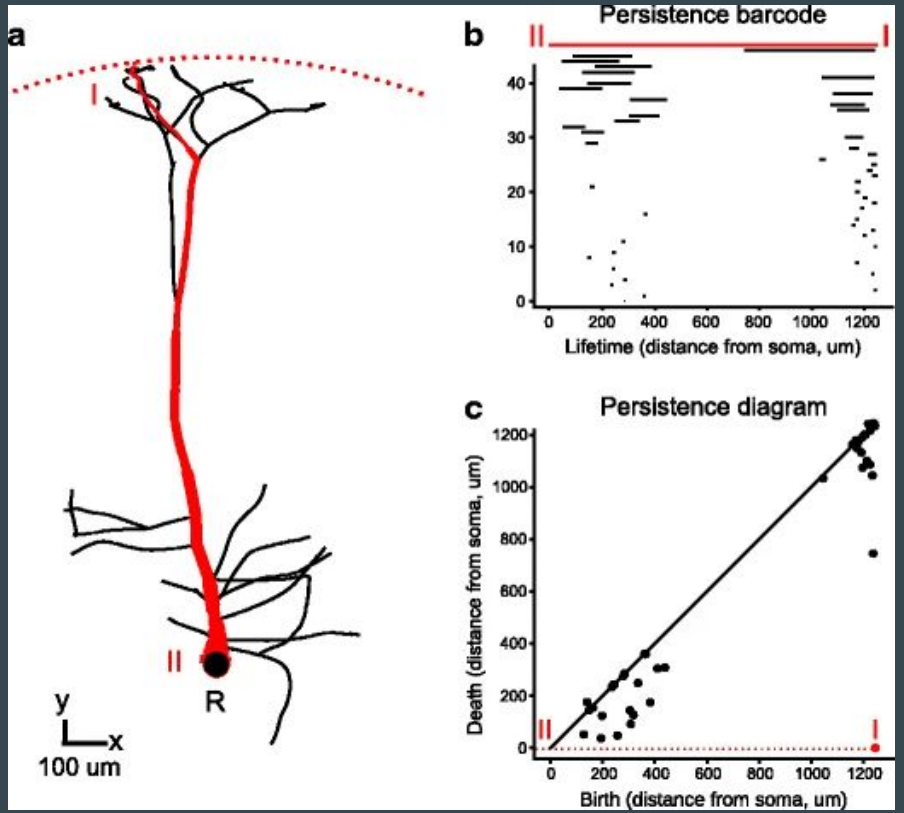
Kanari et al. 2018

# A Topological Representation of Branching Neuronal Morphologies

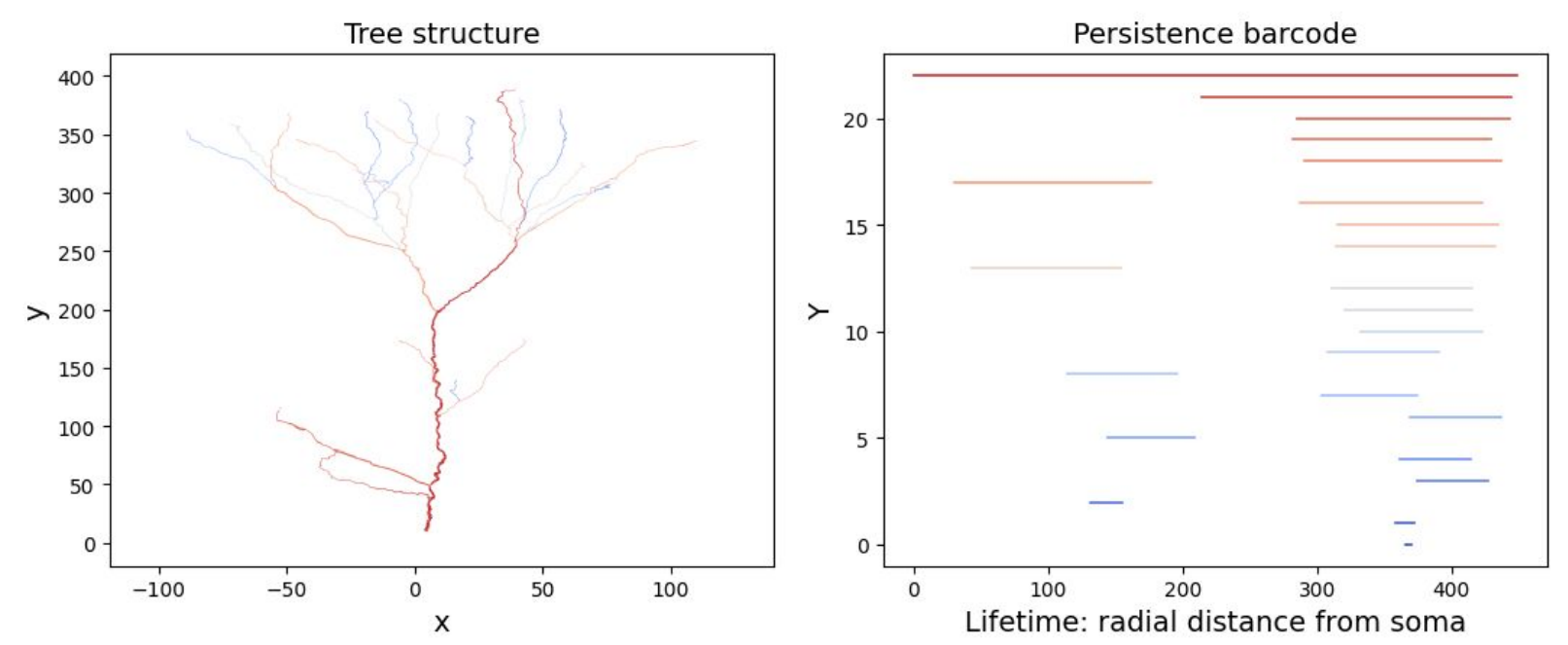
Many biological systems consist of branching structures that exhibit a wide variety of shapes. Our understanding of their systematic roles is hampered from the start by the lack of a fundamental means of standardizing the description of complex branching patterns, such as those of neuronal trees. To solve this problem, we have invented the Topological Morphology Descriptor (TMD), a method for encoding the spatial structure of any tree as a “barcode”, a unique topological signature. As opposed to traditional morphometrics, the TMD couples the topology of the branches with their spatial extents by tracking their topological evolution in 3-dimensional space. We prove that neuronal trees, as well as stochastically generated trees, can be accurately categorized based on their TMD profiles. The TMD retains sufficient global and local information to create an unbiased benchmark test for their categorization and is able to quantify and characterize the structural differences between distinct morphological groups. The use of this mathematically rigorous method will advance our understanding of the anatomy and diversity of branching morphologies.

# Topological Morphology Descriptor

The persistence barcode (B) of a tree (A) represents each component as a horizontal line whose endpoints mark its birth and death in units that depend on the choice of the function f used for the ordering of the nodes of the tree. In our case, it is radial distance of the nodes from the root (R), so the units are microns. The largest component is shown in red together with its birth (I) and death (II). The persistence barcode can be equivalently represented as points in a persistence diagram (C) where the birth (I) and death (II) of a component are the X and Y coordinates of a point respectively (in red).

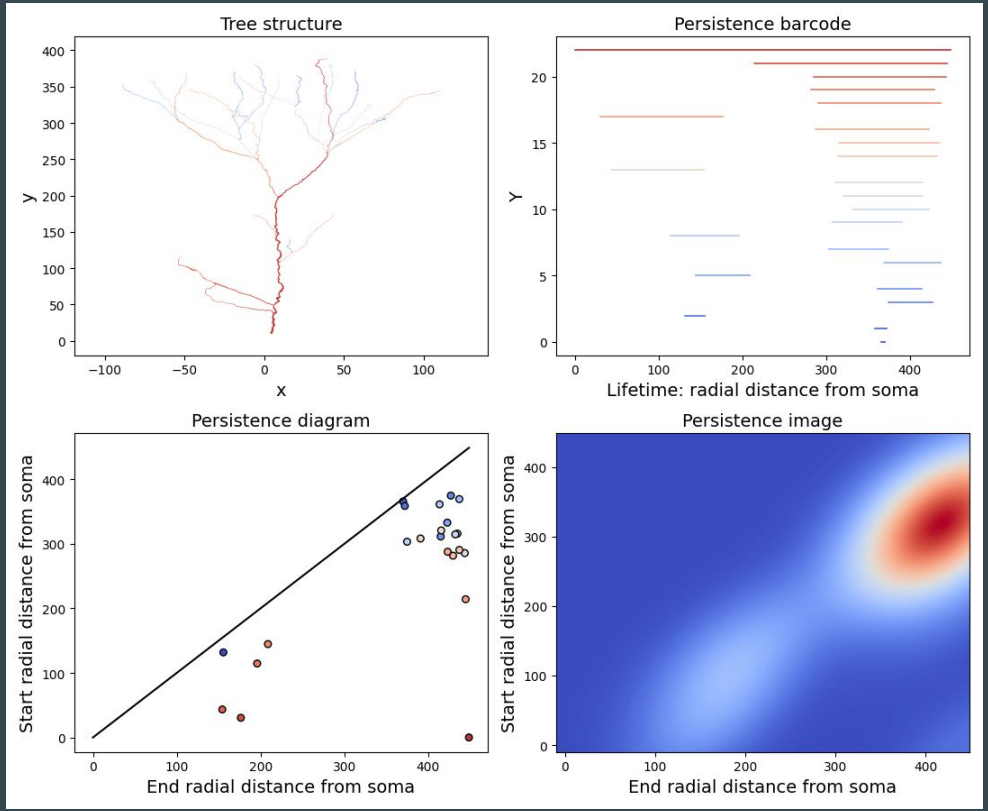


# Topological Morphology Descriptor




Tree decomposition into a barcode from longer (red) to shorter (blue) components

# Topological Morphology Descriptor



# A Topological Representation of Branching Neuronal Morphologies

Given a tree T with vertices $v_i$ and leaves $l_j$, and a function f applied in all vertices $f(v_i)$ the TDM of the tree is given by the following algorithm:

1. Collect all leaves
2. Find all parents of leaves (parent is a vertex one step towards the root)
3. Find all siblings (vertices that share the same parent)
4. Compare their values f, the larger value persists according to Elder rule
5. Repeat the process until the root is reached

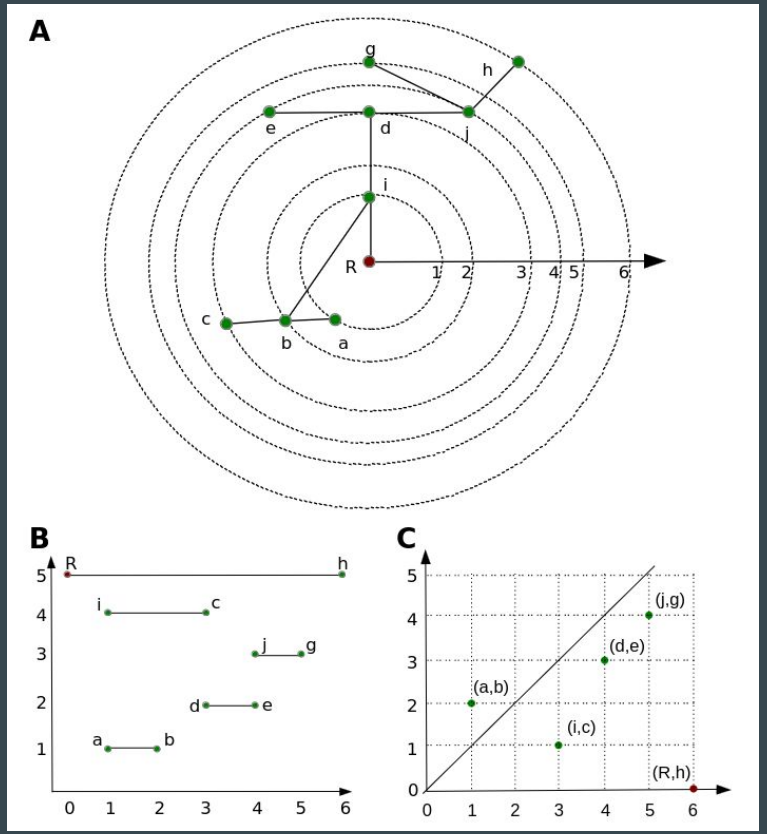
**Algorithm 1** TMD algorithm

**Require:** $T$ with $R, B, L, f : T \to \mathbb{R}$
**Ensure:** TMD$(T, f)$, a persistence barcode obtained from a tree $T$ and the function $f$

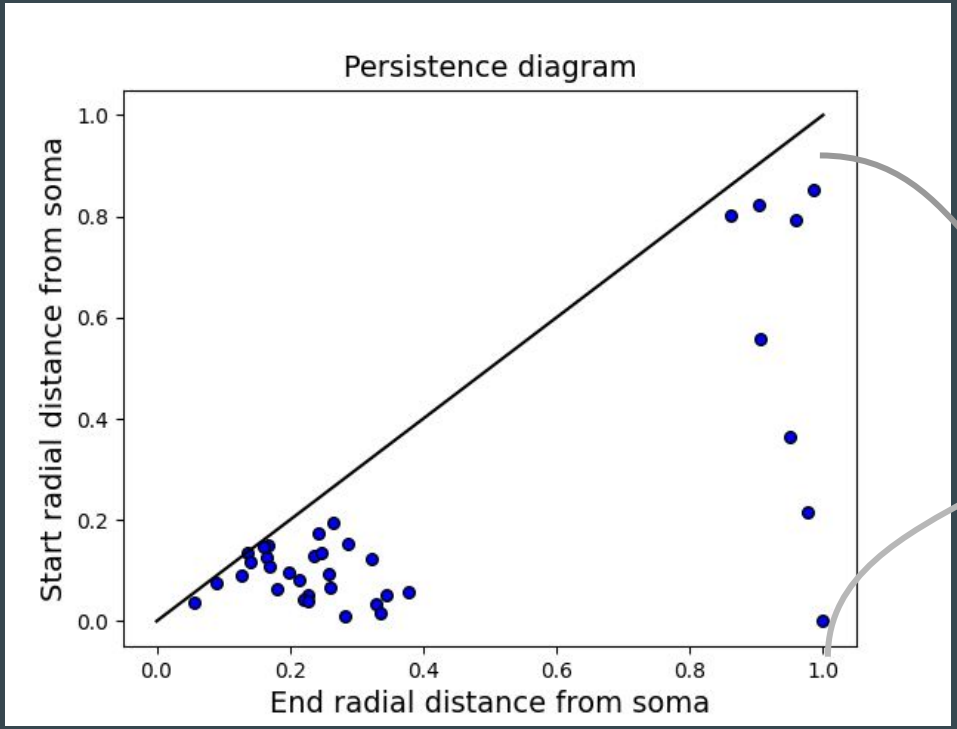
TMD$(T, f)$: empty list to contain pairs of real numbers
$A \leftarrow L$ ▷ $A$ : set of active nodes
**for** every $l \in L$
  $v(l) = f(l)$
**while** $R \notin A$
  **for** $l$ in $A$
    $p$ : parent of $l$
    $C$ : children of $p$
    **if** $\forall n \in C, n \in A$
      $c_m$ : randomly choose one of $\{c \mid v(c) = \max_{c'}(v(c'))$ for $c' \in C\}$
      Add $p$ to $A$
      **for** $c_i$ in $C$
        Remove $c_i$ from $A$
        **if** $c_i \neq c_m$
          Add $(v(c_i), f(p))$ to TMD$(T, f)$
      $v(p) \leftarrow v(c_m)$
Add $(v(R), f(R))$ to TMD$(T, f)$
Return TMD$(T, f)$

# A Topological Representation of Branching Neuronal Morphologies

Demonstration of the TMD algorithm: A simple embedded rooted tree (A) is transformed with the TMD algorithm into the corresponding persistence barcode (B) and the equivalent persistence diagram (C). The root (R) is colored red, while the branch points and leaves are shown in green. The edges connecting corresponding pairs of points are presented by straight lines. The dashed circles are provided as a guide to the eye to indicate different levels of radial distances

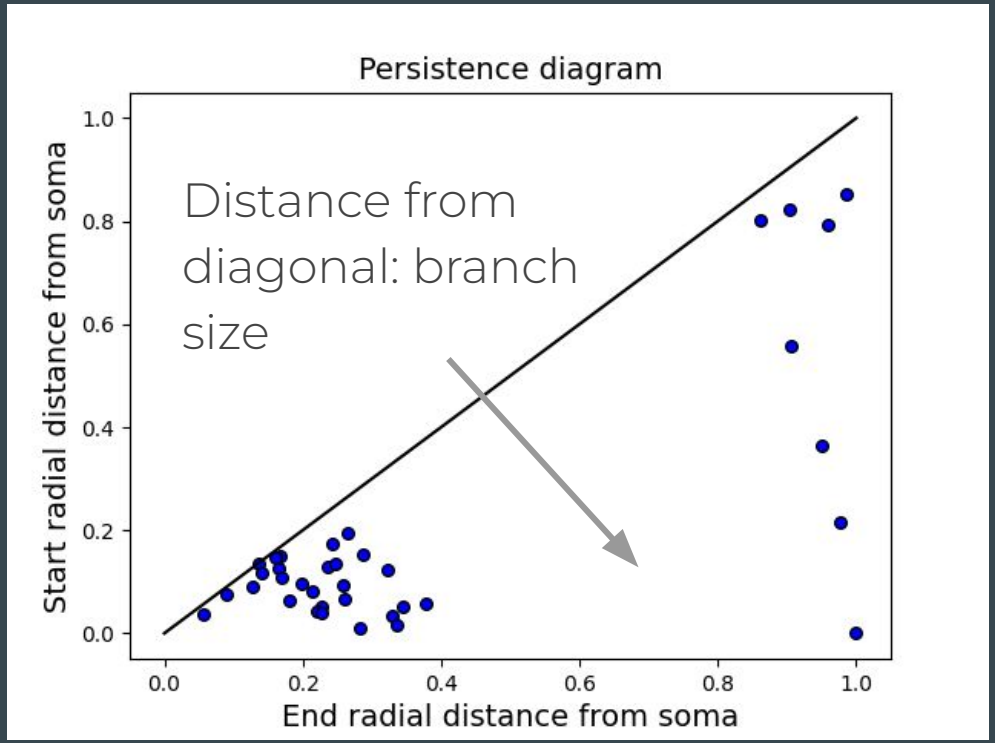


# Topological Morphology Descriptor - Number of branches



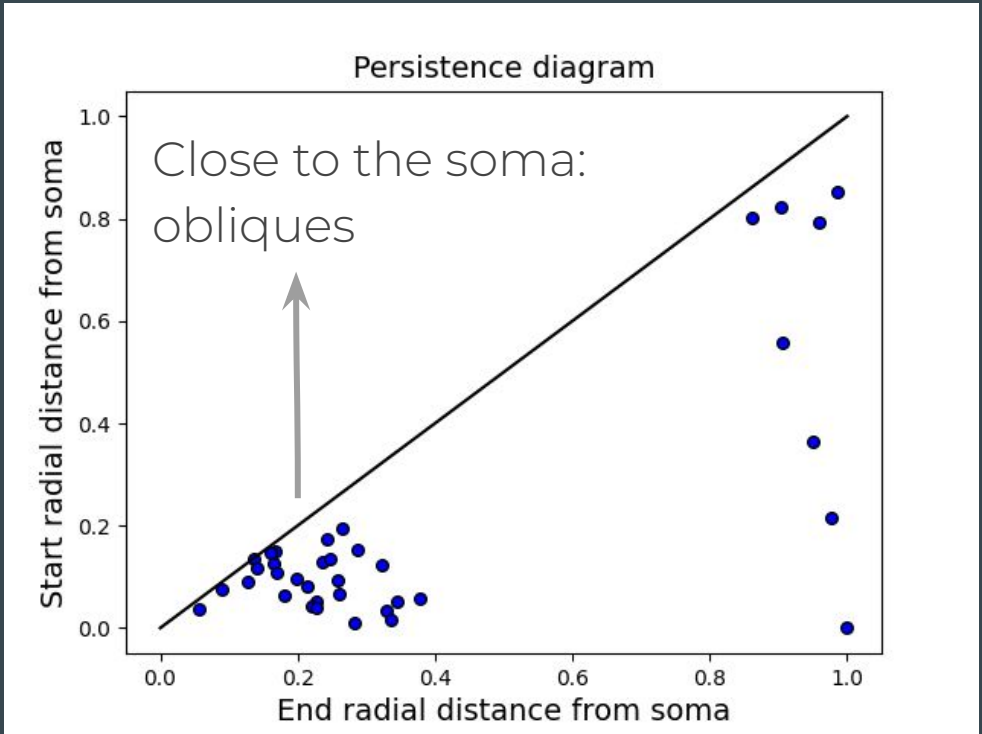

Number of points -> number of branches in the tree

# Topological Morphology Descriptor - Branch length




Far from diagonal: largest branch, corresponds to apical main trunk

# Topological Morphology Descriptor - obliques from tuft

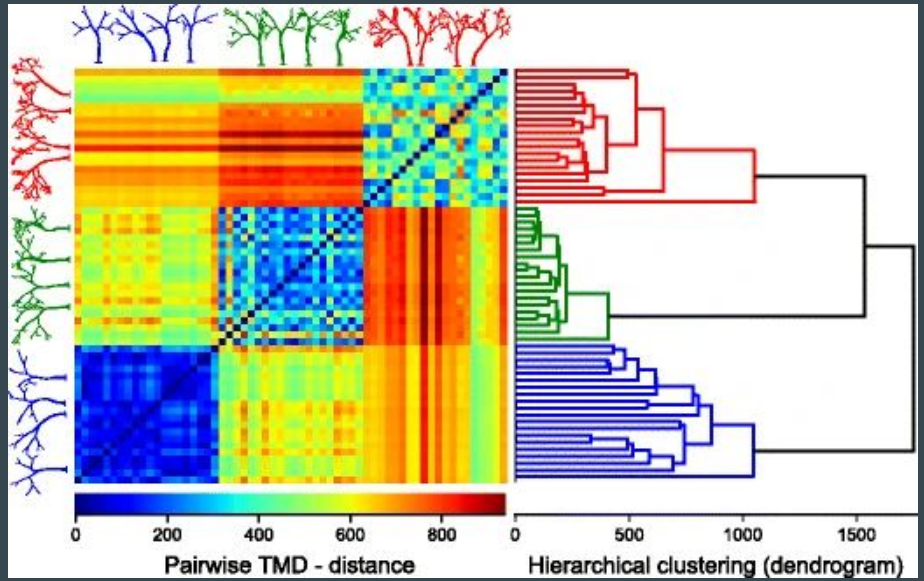



Far from the soma:
apical tufts

# A Topological Representation of Branching Neuronal Morphologies

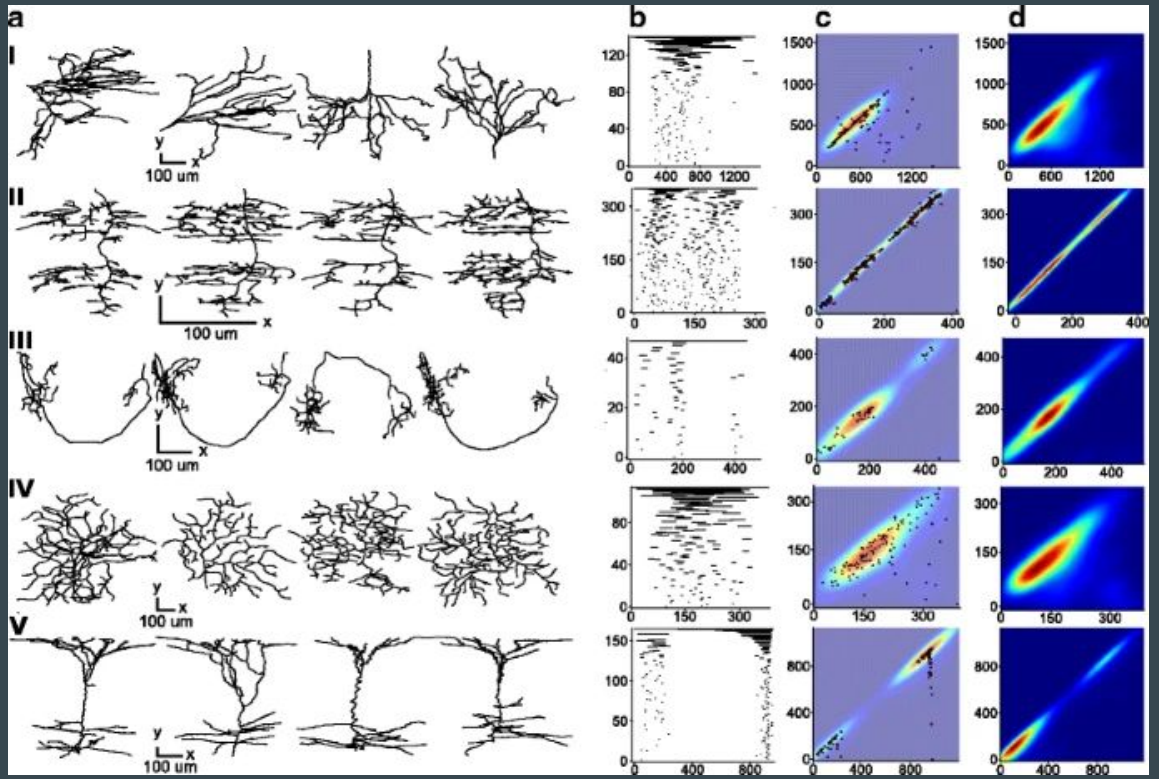
Topological analysis of artificial trees generated using a stochastic process.

Random trees were generated and their betti curve differences (TMD-distance) were used to cluster them. Each group differs from the others only in the tree depth. Each individual of the group is generated using the same tree parameters but a different random number seed. The TMD-distance of the trees allow their accurate separation into groups. The distance matrix indicates the existence of three groups which are identified with high accuracy by a simple dendrogram algorithm.



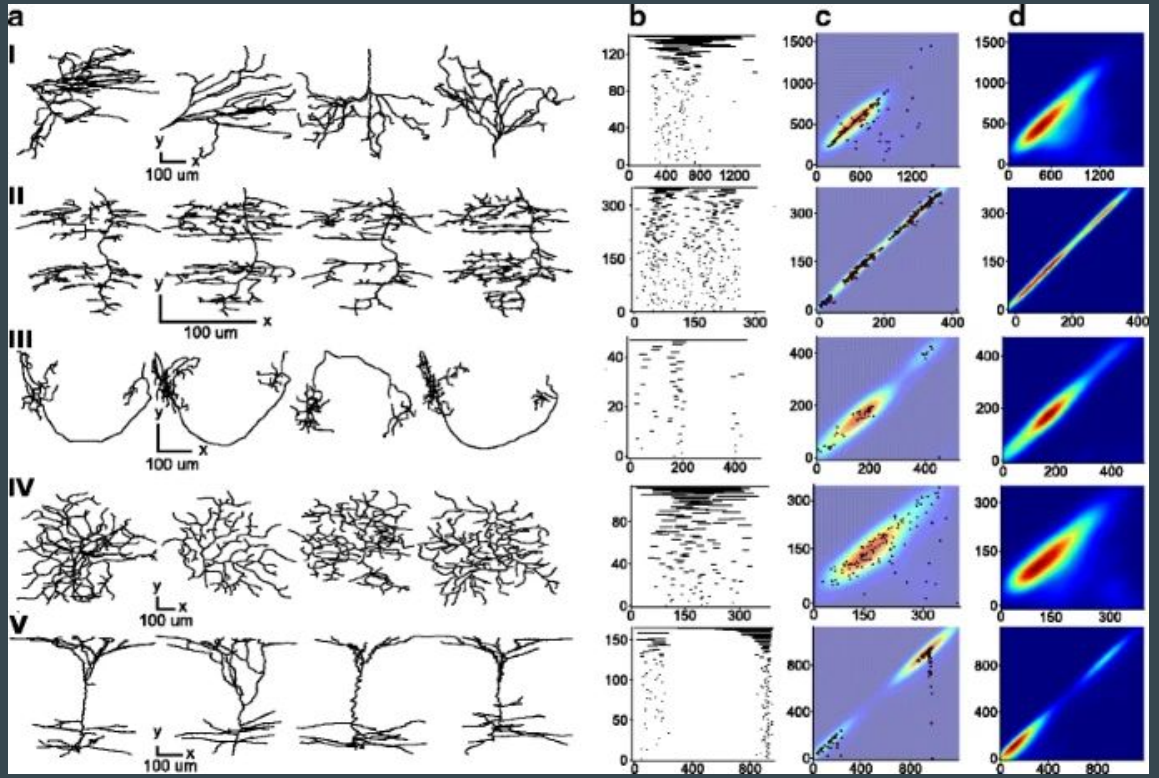
# A Topological Representation of Branching Neuronal Morphologies

Application of topological analysis to a neuronal tree (A) showing the largest persistent component (red). The persistence barcode (B) represents each component as a horizontal line whose endpoints mark its birth an



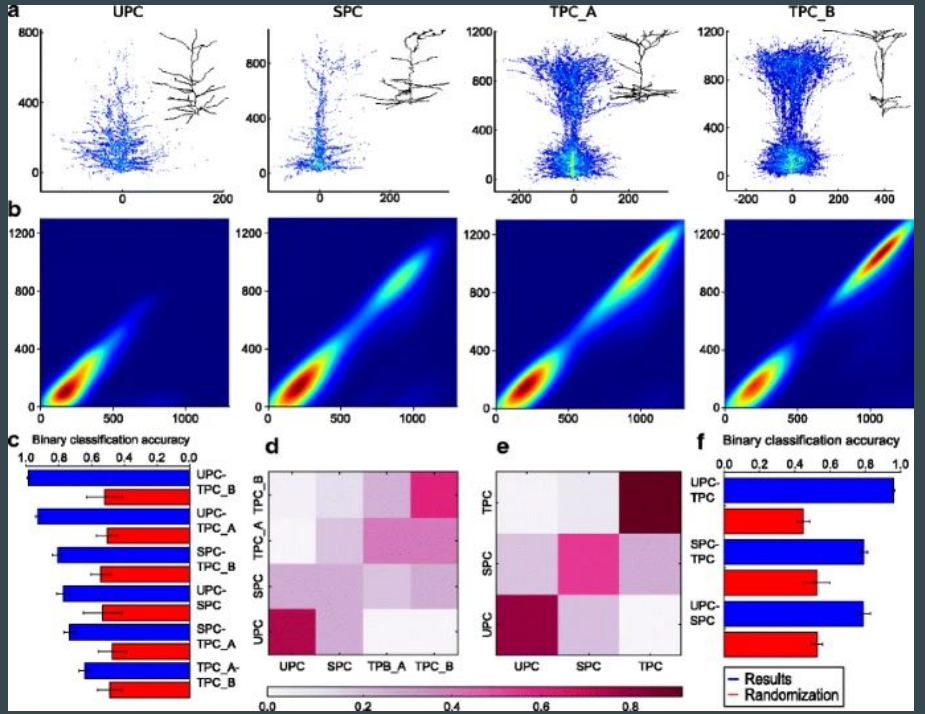
# A Topological Representation of Branching Neuronal Morphologies

The structural differences of the trees are clearly evident in these barcodes. II, III and V have clusters of short components, clearly distinct from the largest component, while I and IV have bars of a quasi-continuous distribution of decreasing lengths. Also, barcodes III, and V indicate the existence of two clusters, while barcodes I and IV are dense overall.



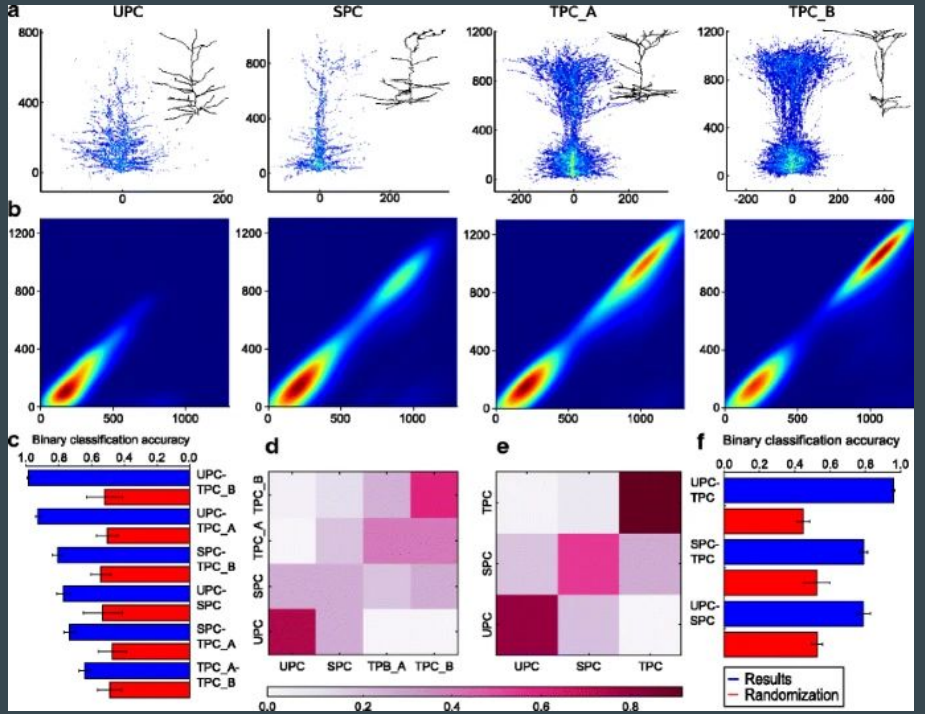
# A Topological Representation of Branching Neuronal Morphologies

Comparison of the TMD of apical dendrite trees extracted from several types of rat pyramidal neuron. Four cell types are shown in (A): UPC, SPC, TPC-A, TPC-B (left to right). The morphological differences between these cell types are subtle, but the unweighted persistence images (B) clearly reveal them, particularly the presence of two clusters in the TPC-A and TPC-B cell types.



# A Topological Representation of Branching Neuronal Morphologies

From these unweighted persistence images we train a decision tree classifier on the expert-assigned groups of cells. The binary classification (C) and the confusion matrix (D) based on the TMD algorithm shows an overlap of TPC-A and TPC-B trees. When those two classes are merged (E, F) the separation between the remaining types is evident. This result shows that the unweighted persistence images objectively support the expert's classification when the morphological differences between the classes are significant

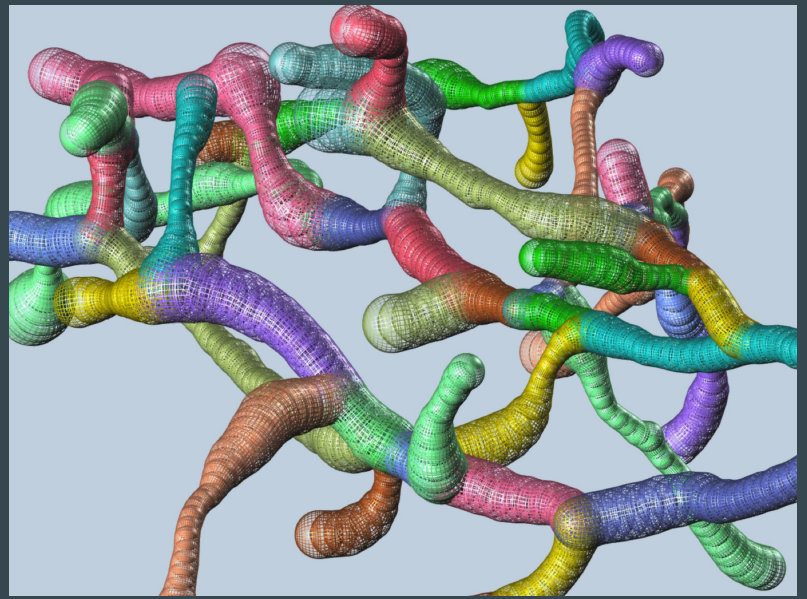


# A Topological Representation of Branching Neuronal Morphologies

Comparison of the TMD of BigNeuron neuronal morphologies.

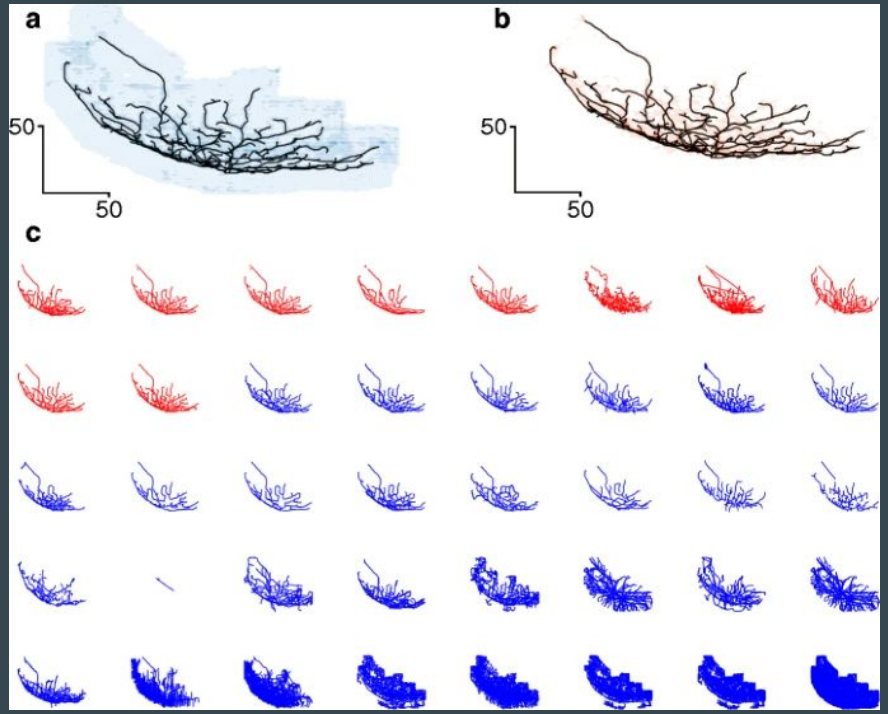
Big Neuron is an effort by Allen Institute (Seattle) to create automatic reconstruction algorithms.

Their automatic algorithms generate hundreds of automatic reconstructions. The question is how to assess quality of each algorithm.



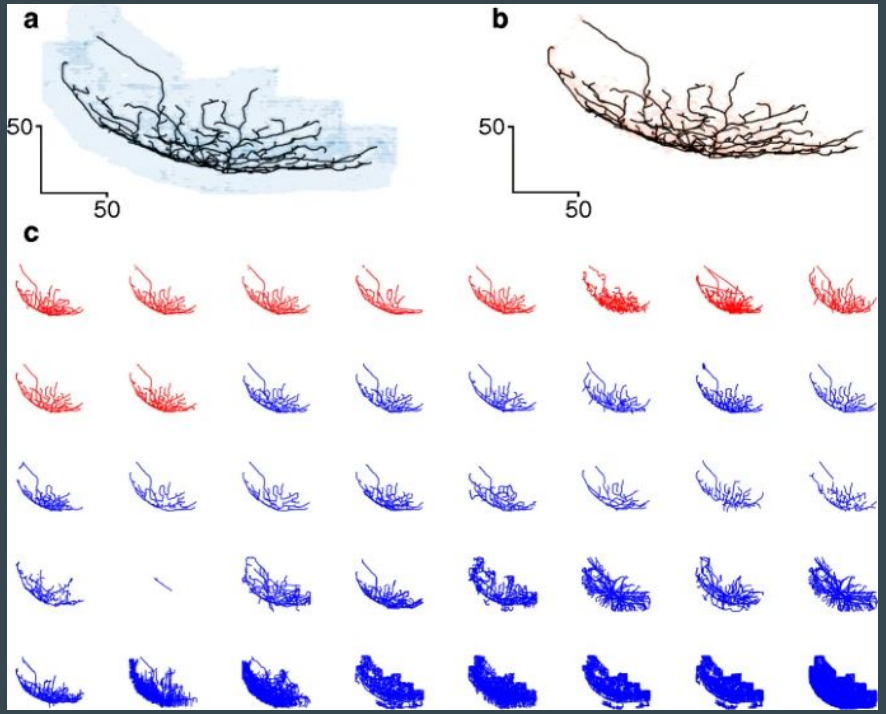
# A Topological Representation of Branching Neuronal Morphologies

Comparison of the TMD of BigNeuron neuronal morphologies. An image stack is used for the manual reconstruction (reference neuron) and for the automatic reconstructions produced by a variety of community supplied algorithms. The results of each algorithm are illustrated in panel C, from best (top left) to worst (bottom right).



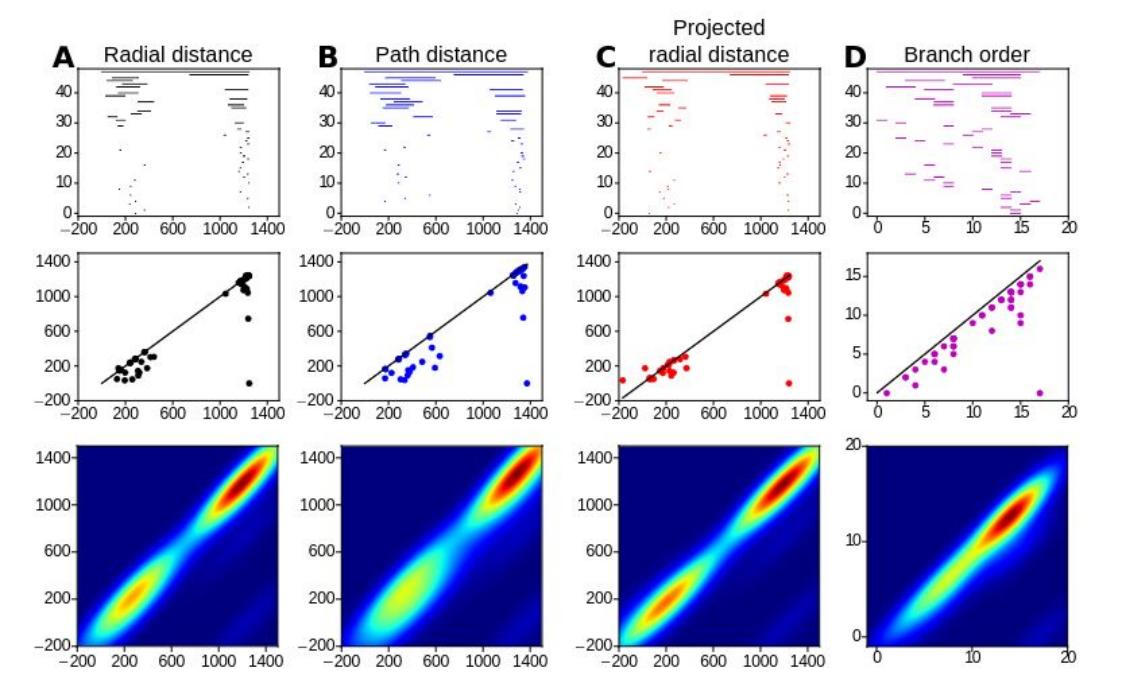
# A Topological Representation of Branching Neuronal Morphologies

The reference neuron (in black) is visualized against the density plot of all the automatically reconstructed neurons (A, in blue), and the density plot of the ten best automatic reconstructions (B, in red), ranked according to their TMD-distance from the reference neuron. A comparison between panels A and B shows that the density plot of the ten highest ranked automatic reconstructions closely matches the structure of the reference morphology



# A Topological Representation of Branching Neuronal Morphologies

Demonstration of TMD algorithm for different morphological features. A. Radial distance from the soma. B. Path distance from the soma. C. Projected radial distance from the soma to the axis normal to the pia; this measurement can discriminate trees with different spatial distributions. D. Branch order; this measurement does not take into account the embedding in space, only the combinatorial branching patterns of the tree.



# Objective Morphological Classification of Neocortical Pyramidal Cells

Kanari et al. 2019

# Objective Morphological Classification of Neocortical PCs

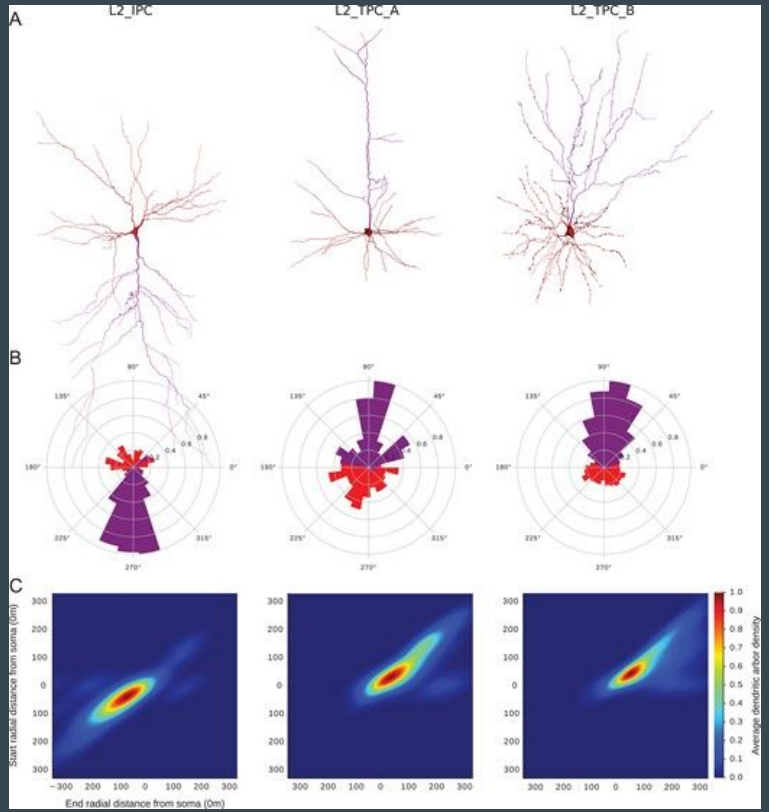
Many biological systems consist of branching structures that exhibit a wide variety of shapes. Our understanding of their systematic roles is hampered from the start by the lack of a fundamental means of standardizing the description of complex branching patterns, such as those of neuronal trees. To solve this problem, we have invented the Topological Morphology Descriptor (TMD), a method for encoding the spatial structure of any tree as a “barcode”, a unique topological signature. As opposed to traditional morphometrics, the TMD couples the topology of the branches with their spatial extents by tracking their topological evolution in 3-dimensional space. We prove that neuronal trees, as well as stochastically generated trees, can be accurately categorized based on their TMD profiles. The TMD retains sufficient global and local information to create an unbiased benchmark test for their categorization and is able to quantify and characterize the structural differences between distinct morphological groups. The use of this mathematically rigorous method will advance our understanding of the anatomy and diversity of branching morphologies.

# Objective Morphological Classification of Neocortical PCs

Three PC types/subtypes in Layer 2.

(A) Exemplar reconstructed morphologies of PC dendrites: the apical dendrite is presented in purple and the basal dendrites in red.

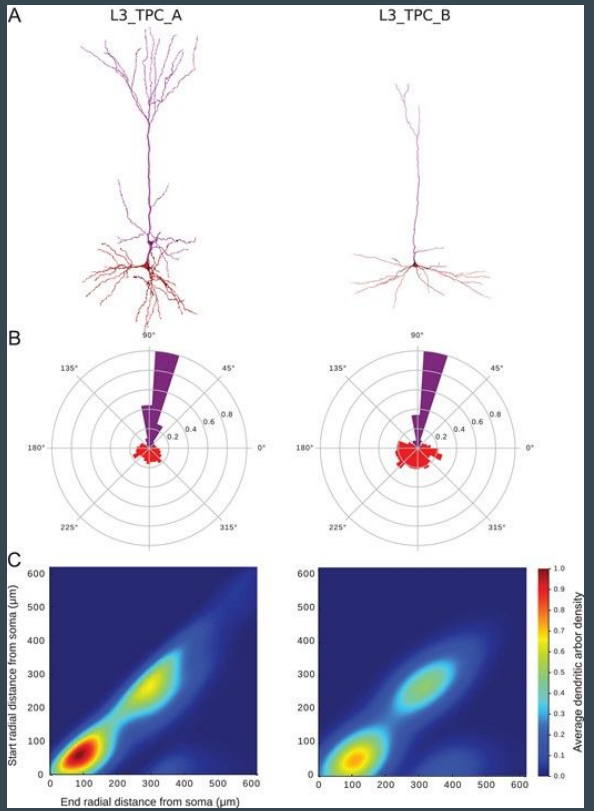
(B) Polar plot analysis of dendritic branches (apical in purple, basal in red). Tufted PCs are oriented towards the pia and the inverted PCs in the opposite direction as they project towards the white matter.



# Objective Morphological Classification of Neocortical PCs

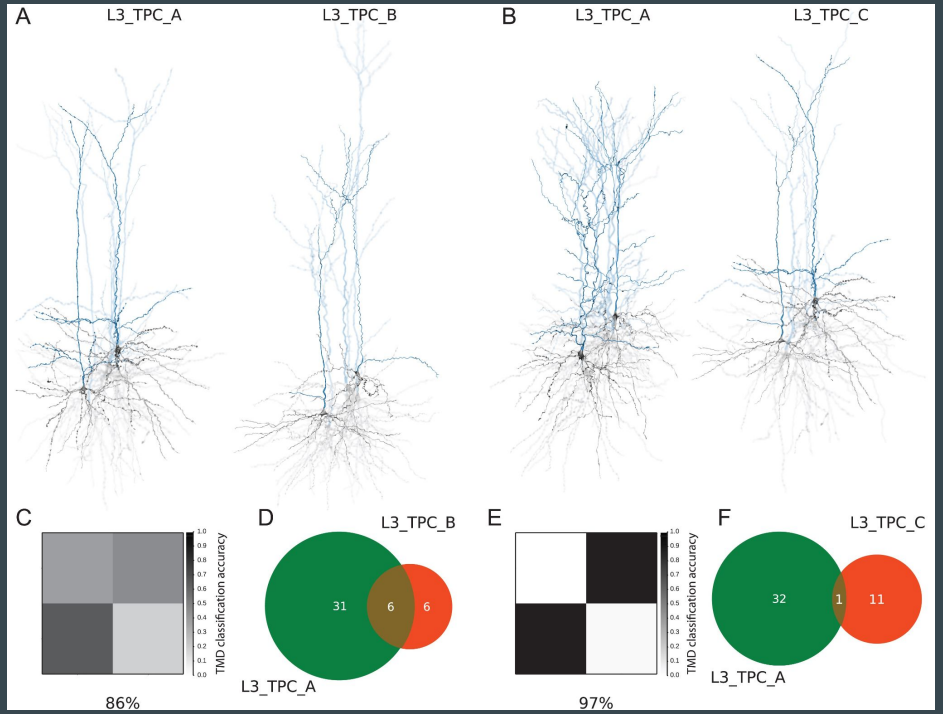
Two PC types/subtypes in Layer 3.

Similar separation of L3 neurons (only two classes makes the distinction between them easier)



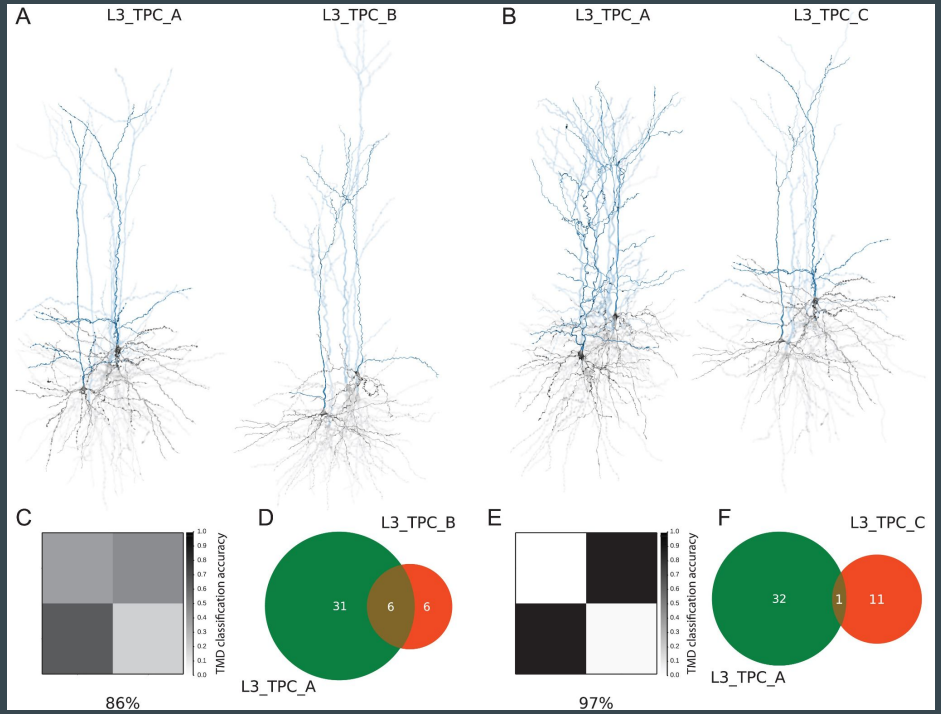
# Objective Morphological Classification of Neocortical PCs

Reclassification of Layer 3 PCs. (A) Curated renderings of L3_TPC_A and L3_TPC_B selected morphologies as proposed by expert classification. (B) Curated renderings of L3_TPC_A and L3_TPC_B selected morphologies, after TMD-based reclassification.



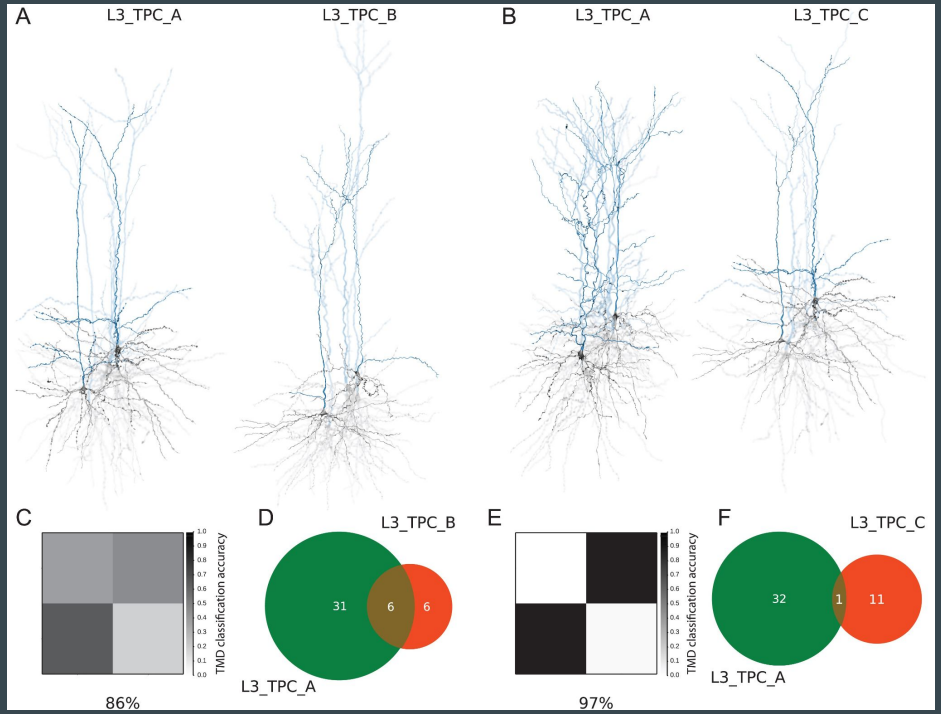
# Objective Morphological Classification of Neocortical PCs

(C) The confusion matrix illustrates the large percentage of misclassified cells between the expert proposed subtypes, yielding a total accuracy of 86%. (D) The 2 subtypes are usually misclassified, as half of the L3_TPC_B are confused as L3_TPC_A.



# Objective Morphological Classification of Neocortical PCs

(E) The confusion matrix illustrates the clear separation of the 2 subtypes after the TMD-based reclassification and the improved accuracy of the classifier (97%). (F) The 2 subtypes are rarely misclassified, as almost all (~98%) of cells are unambiguously assigned into the 2 subtypes.



# Objective Morphological Classification of Neocortical PCs

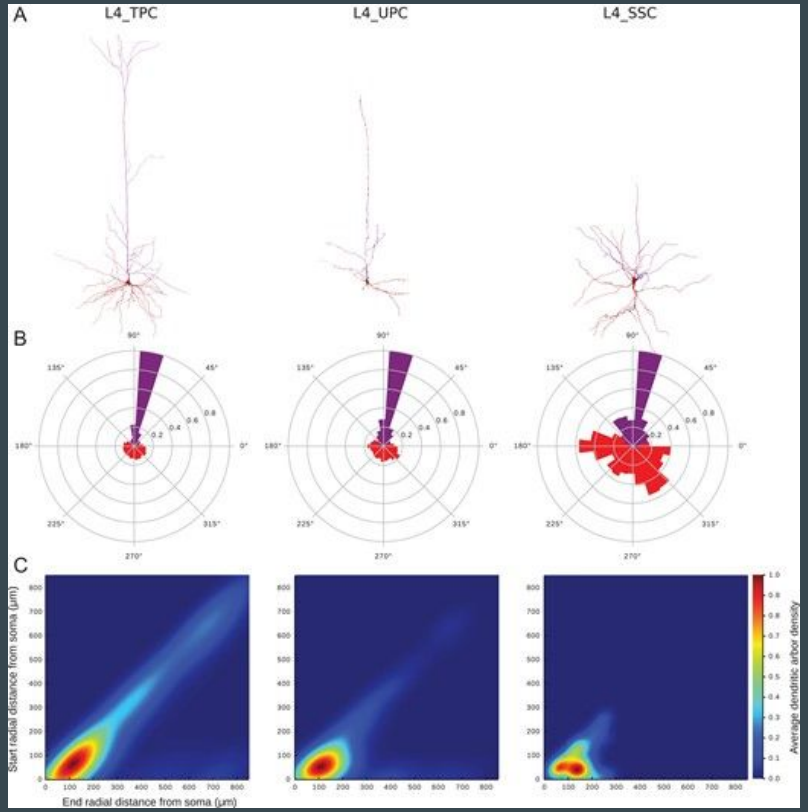
Three PC types/subtypes in Layer 4.

Clearly separated based on their branching structure:

TPC: tufted

UPC: untufted

SSC: smaller - non apical like trees



# Objective Morphological Classification of Neocortical PCs

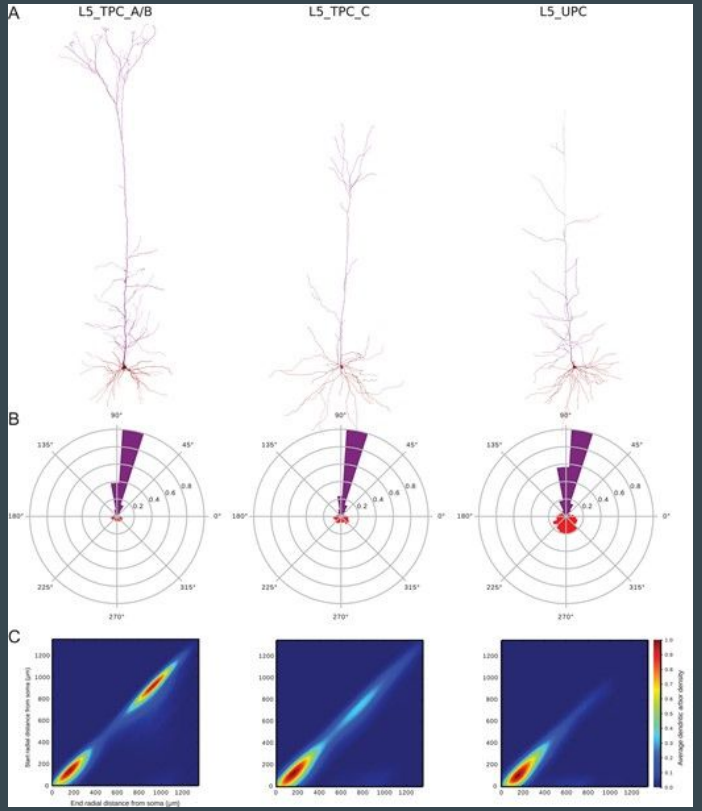
Three PC types/subtypes in Layer 5.

Clearly separated based on their branching structure:

TPC (A,B): thick tufted

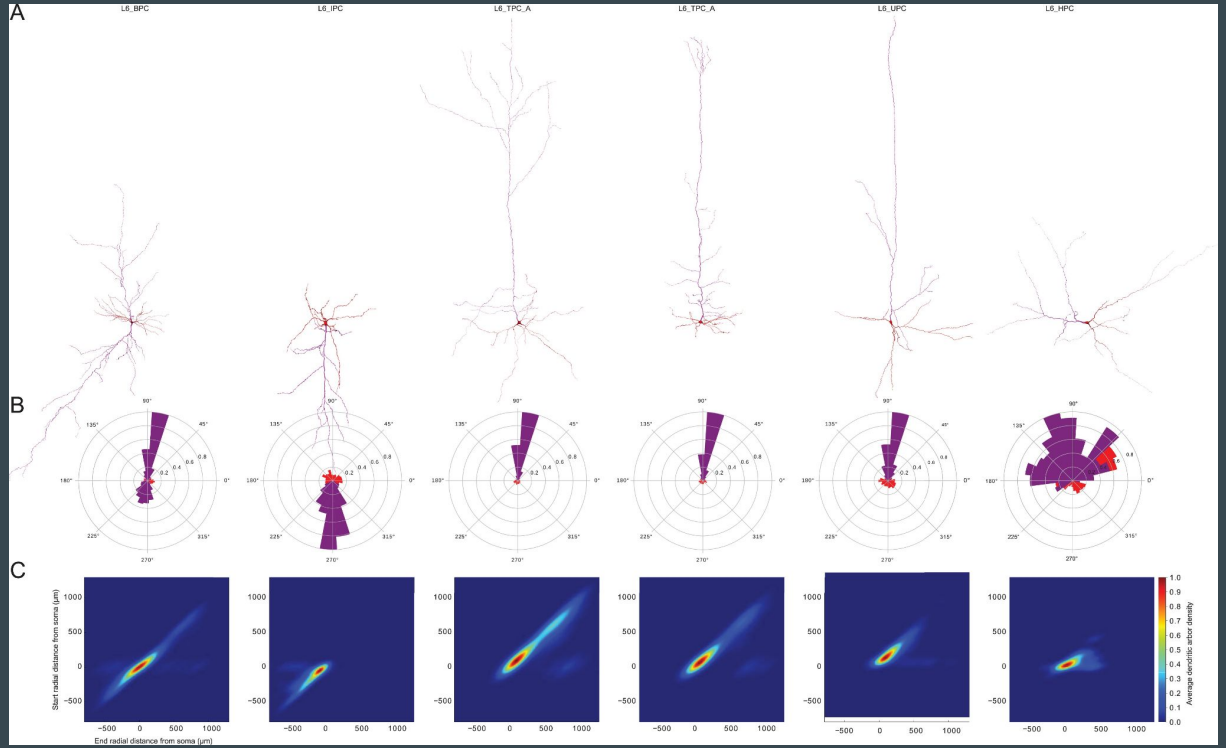
TPC (C): small tufted

UPC: untufted



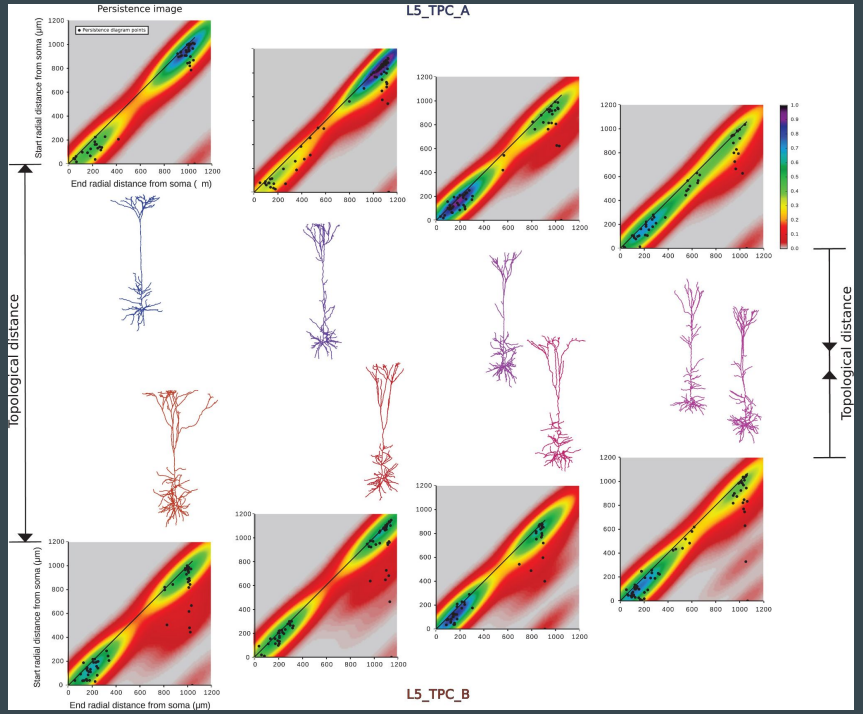
# Objective Morphological Classification of Neocortical PCs

Six PC types/subtypes in Layer 6. More complicated for deeper layers:



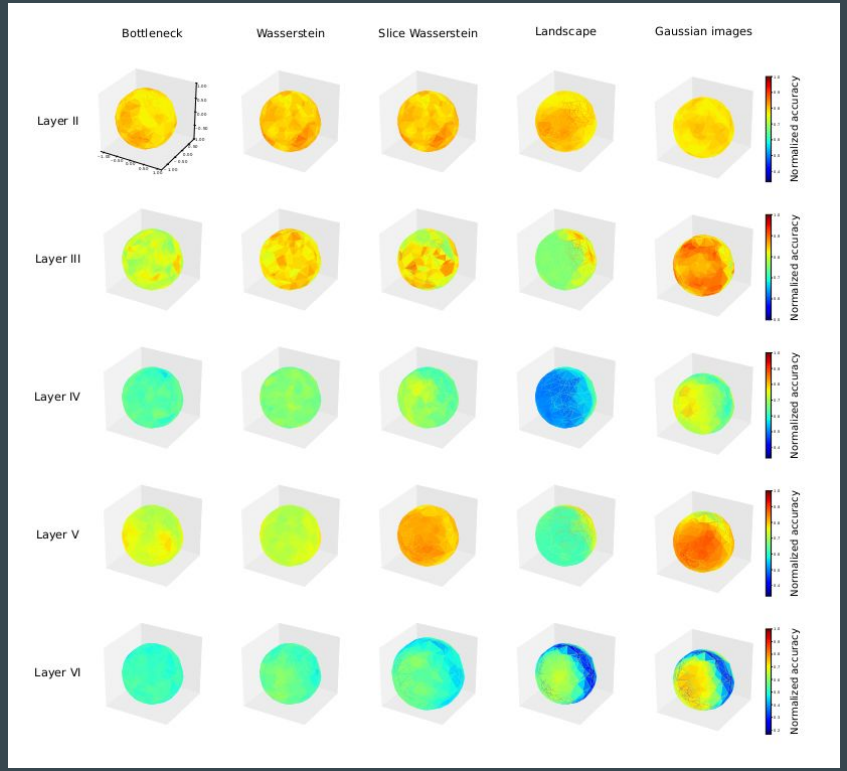
# Objective Morphological Classification of Neocortical PCs

Convergence of subtypes L5TPC_A and B. Illustration of selected dendritic morphologies of L5_TPC_A (in blue) and L5_TPC_B (in red) of decreasing topological distance (from left to right). For border cases the 2 subtypes are very well separated (extreme left). The persistence images of all the presented apical trees are shown, and the points of the persistence diagrams for each apical tree are superimposed on the respective persistence images. However, as the topological distance decreases as the persistence images converge (left to right), and morphologies exhibit similar topological shapes (extreme right).



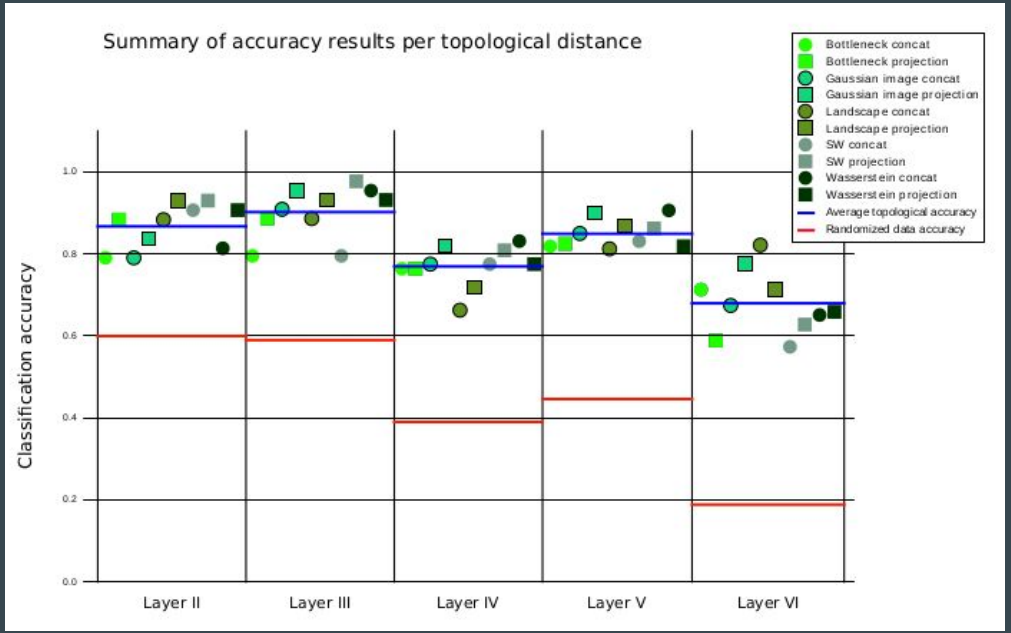
# Objective Morphological Classification of Neocortical PCs

Accuracy of TMD – classification based on five topological distances. The final proposed classification for the PCs of each Layer is tested with five different topological distances (from left to right: Bottle- neck, Wasserstein, Slice-Wasserstein, Landscape, Persistence images). For each distance, the accuracy of the classification for 300 different orientations is computed and visualized on a unit sphere. The colormap illustrates the accuracy of the classification per Layer, where the base-line is defined by the accuracy of a randomized labeling



# Objective Morphological Classification of Neocortical PCs

Accuracy of TMD classification and comparison of different topological distances. For each layer the accuracy of the classification based on five different topological distances (Bottleneck, Persistence images, Landscape, Slice Wasserstein, Wasserstein) is computed for 300 different orientations. The best projection (square) and the summed kernel of all projections (circle) are presented with different shades of green.



# Questions?