Topological data analysis and persistent homology

Developing an outlier detector based on persistent homology

Arnau Mas

Supervised by
Dr Albert Ruiz

Universitat Autònoma de Barcelona
September 3rd 2020

Universitat Autònoma
de Barcelona

Outline

What is radiomics?

- Has its origins in the field of oncology: tumour imaging

What is radiomics?

- Has its origins in the field of oncology: tumour imaging
- Extract many radiomic features from scans

What is radiomics?

- Has its origins in the field of oncology: tumour imaging
- Extract many radiomic features from scans
- Look for correlations between radiomic features and treatment response

What is radiomics?

UAB

Benefits

- Non-invasive
- More systematic
- Quantitative

What is radiomics?

UAB

Benefits

- Non-invasive
- More systematic
- Quantitative

Drawbacks

- May be hard to reproduce
- High dimensional feature spaces
- Limited number of cases

- TOPiomics is an ongoing collaboration between the Computer Vision Center (CVC) and the Vall d'Hebron Institute of Oncology (VHIO).

- TOPiomics is an ongoing collaboration between the Computer Vision Center (CVC) and the Vall d'Hebron Institute of Oncology (VHIO).
- Main aim: early detection of patients with strange radiomic signatures, *outliers*, for whom standard treatments could fail.

- TOPiomics is an ongoing collaboration between the Computer Vision Center (CVC) and the Vall d'Hebron Institute of Oncology (VHIO).
- Main aim: early detection of patients with strange radiomic signatures, *outliers*, for whom standard treatments could fail.
- Use *topological data analysis*, hence TOPiomics (topological radiomics).

Prior work

- Progress thus far: use the *mutual k-nearest neighbours* (MkNN) graph to detect outliers.

Prior work

- Progress thus far: use the *mutual k-nearest neighbours* (MkNN) graph to detect outliers.
- Problem: there is a parameter that has to be fixed. It has to do with the scale at which we look at the data.

Prior work

- Progress thus far: use the *mutual k-nearest neighbours* (MkNN) graph to detect outliers.
- Problem: there is a parameter that has to be fixed. It has to do with the scale at which we look at the data.
- Idea: look at *every* parameter value.

Prior work

- Progress thus far: use the *mutual k-nearest neighbours* (MkNN) graph to detect outliers.
- Problem: there is a parameter that has to be fixed. It has to do with the scale at which we look at the data.
- Idea: look at *every* parameter value.

This is persistence

Outline

The elevator pitch

Algebraic topology studies topological spaces by computing algebraic invariants, namely with functors

$$\mathrm{Top} \to \mathrm{Grp}$$

$$\mathrm{Top} \to \mathrm{Ab}$$

$$\mathrm{Top} \to \mathrm{Ring}$$

$$\mathrm{Top} \to \mathrm{Vect}_K$$

$$\vdots$$

The elevator pitch

Algebraic topology studies topological spaces by computing algebraic invariants, namely with functors

$$\begin{aligned} \text{Top} &\to \text{Grp} \\ \text{Top} &\to \text{Ab} \\ \text{Top} &\to \text{Ring} \\ \text{Top} &\to \text{Vect}_K \\ &\vdots \end{aligned}$$

Homology groups are one of these functors,

$$H_n \colon \text{Top} \to \text{Ab}.$$

The elevator pitch

Algebraic topology studies topological spaces by computing algebraic invariants, namely with functors

$$\begin{gathered}\mathrm{Top} \to \mathrm{Grp} \\ \mathrm{Top} \to \mathrm{Ab} \\ \mathrm{Top} \to \mathrm{Ring} \\ \mathrm{Top} \to \mathrm{Vect}_K \\ \vdots\end{gathered}$$

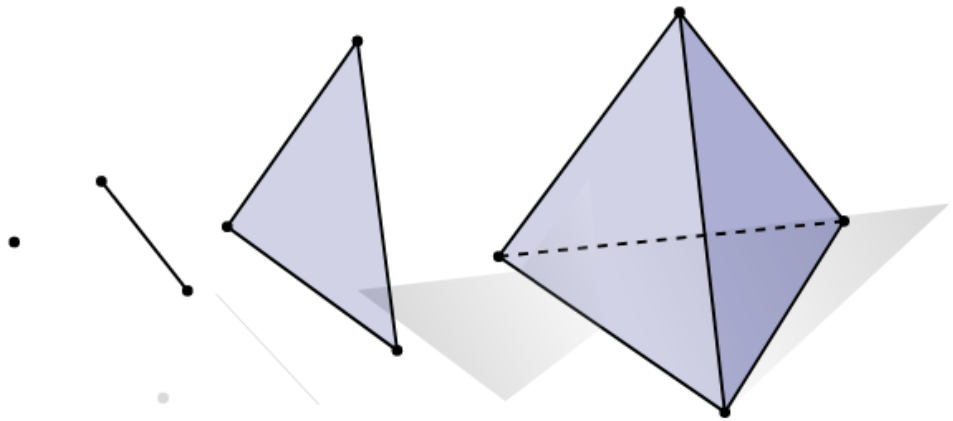
Homology groups are one of these functors,

$$H_n \colon \mathrm{Top} \to \mathrm{Ab}.$$

They come in many flavours, the simplest one is *simplicial homology*, which works for *simplicial complexes*.

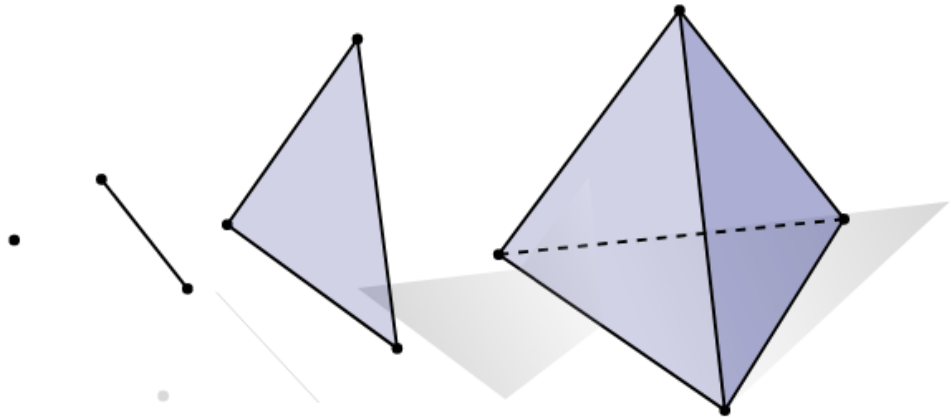
The ingredients of homology

- *Simplex*. The convex hull of a set of points.



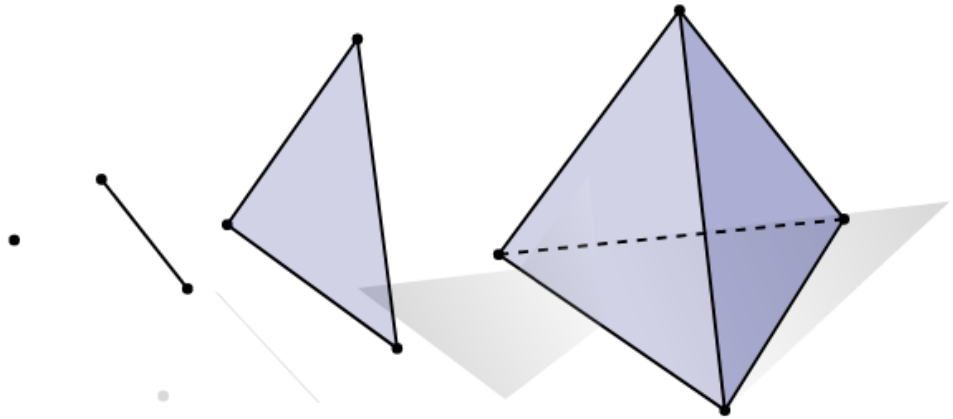
The ingredients of homology

- *Simplex*. The convex hull of a set of (geometrically independent) points.



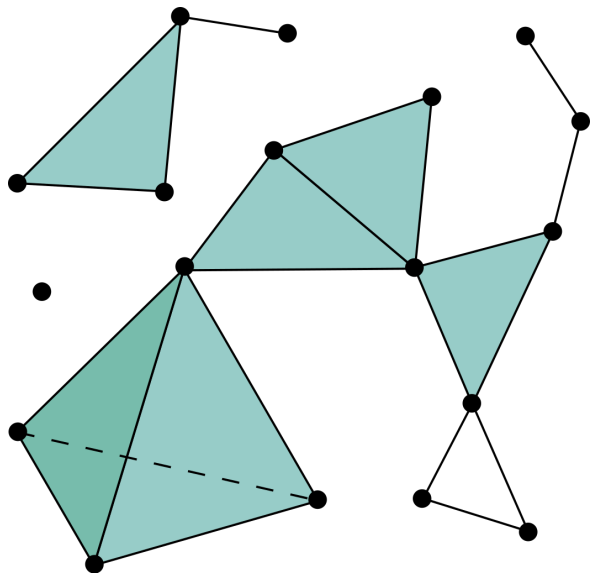
The ingredients of homology

- *Simplex*. The convex hull of an (ordered) set of (geometrically independent) points.



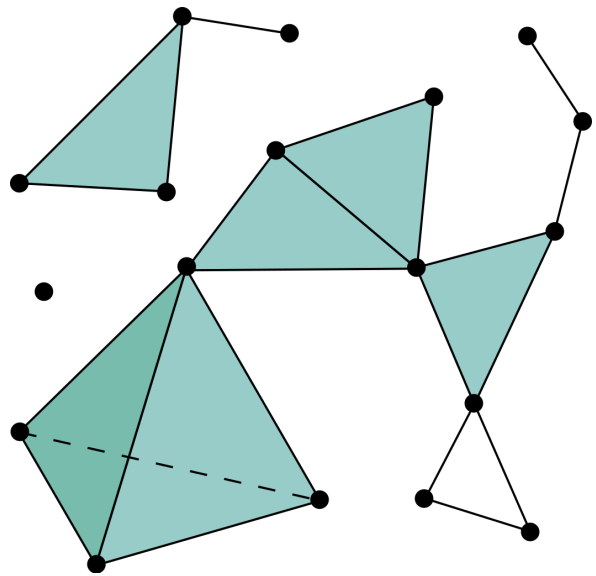
The ingredients of homology

- *Simplex*. The convex hull of an (ordered) set of (geometrically independent) points.
- *Simplicial complex*, K.
  Topological space assembled out of simplices glued along their faces.



The ingredients of homology

- *Simplex*. The convex hull of an (ordered) set of (geometrically independent) points.
- *Simplicial complex*, K. Topological space assembled out of simplices glued along their faces.
- *Chain groups*, $C_n(K)$. Free abelian groups generated by the n-simplices of a complex.

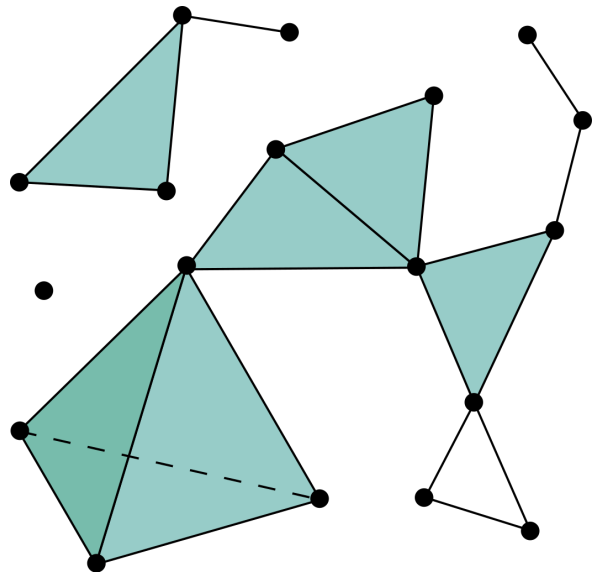


Remark

In the combinatorial picture, the geometrical requirements are dropped and one speaks of *abstract* simplices, *abstract* simplicial complexes, etc.

The ingredients of homology

- *Simplex*. The convex hull of an (ordered) set of (geometrically independent) points.
- *Simplicial complex*, K. Topological space assembled out of simplices glued along their faces.
- *Chain groups*, $C_n(K)$. Free abelian groups generated by the n-simplices of a complex.



Remark

Free abelian groups are equivalent to $\mathbb{Z}$-modules. Alternatively, one can also consider the vector spaces generated by the chains, for instance over $\mathbb{F}_2$.

Orientation

The simplex generated by $p_0, \ldots, p_n$ is written $[p_0, \ldots, p_n]$. The ordering determines an orientation, and we require that flipping orientation implies a change of sign in $C_n(K)$, e.g.

$$[p_0, p_1, \ldots, p_n] = -[p_1, p_0, \ldots, p_n].$$

The boundary morphism

Orientation

The simplex generated by $p_0, \ldots, p_n$ is written $[p_0, \ldots, p_n]$. The ordering determines an orientation, and we require that flipping orientation implies a change of sign in $C_n(K)$, e.g.

$$[p_0, p_1, \ldots, p_n] = -[p_1, p_0, \ldots, p_n].$$

The chain groups are not really free!

The boundary morphisms

Define the boundary morphism $\partial_n \colon C_n(K) \to C_{n-1}(K)$ by

$$\partial_n[p_0, \ldots, p_n] := \sum_{k=0}^{n} (-1)^k [p_0, \ldots, \hat{p}_k, \ldots, p_n]$$

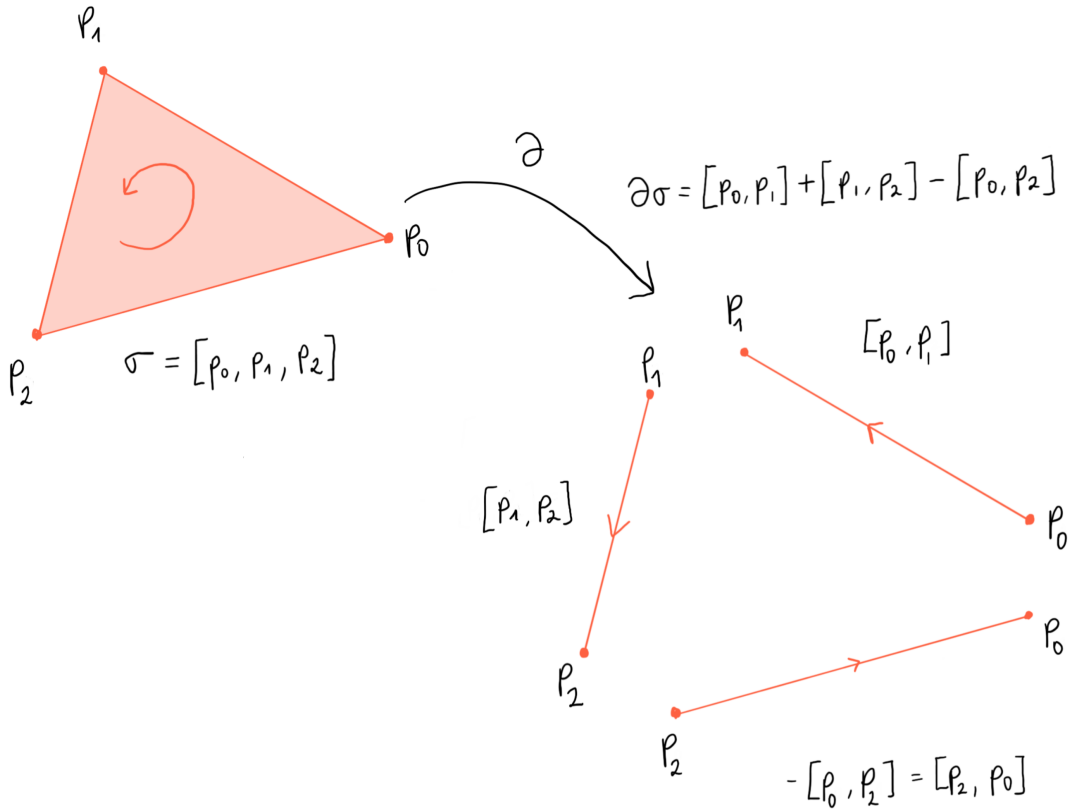
and extending to chains.

The boundary morphisms

Define the boundary morphism $\partial_n \colon C_n(K) \to C_{n-1}(K)$ by

$$\partial_n[p_0, \ldots, p_n] := \sum_{k=0}^{n} (-1)^k [p_0, \ldots, \hat{p}_k, \ldots, p_n]$$

and extending to chains.



Boundaries and cycles

- *Cycles:* chains with no boundary,

$$Z_n(K) := \ker \partial_n \subseteq C_n(K)$$

Boundaries and cycles

- *Cycles:* chains with no boundary,

$$Z_n(K) := \ker \partial_n \subseteq C_n(K)$$

- *Boundaries:* chains which are the boundary of a higher-dimensional chain,

$$B_n(K) := \operatorname{im} \partial_{n+1} \subseteq C_n(K)$$

Boundaries and cycles

- *Cycles:* chains with no boundary,

$$Z_n(K) := \ker \partial_n \subseteq C_n(K)$$

- *Boundaries:* chains which are the boundary of a higher-dimensional chain,

$$B_n(K) := \operatorname{im} \partial_{n+1} \subseteq C_n(K)$$

Fact

$$\partial_n \circ \partial_{n+1} = 0$$

so boundaries are always cycles.

Homology groups

Homology groups are the quotients

$$H_n(K) := Z_n(K)/B_n(K).$$

Homology groups

Homology groups are the quotients

$$H_n(K) := Z_n(K)/B_n(K).$$

The key insight: Homology groups measure cycles which are not boundaries.

Homology groups

Homology groups are the quotients

$$H_n(K) := Z_n(K)/B_n(K).$$

The key insight: Homology groups measure cycles which are not boundaries. They tell us about the number of n-dimensional holes in our space.

Homology groups

Homology groups are the quotients

$$H_n(K) := Z_n(K)/B_n(K).$$

The key insight: Homology groups measure cycles which are not boundaries. They tell us about the number of n-dimensional holes in our space.
Homology is invariant under homotopy, weaker than invariance under homoemorphism.

What is persistence?

Instead of a single simplicial complex, consider a *filtration*,

$$K^0 \subseteq K^1 \subseteq \cdots \subseteq K^N.$$

What is persistence?

Instead of a single simplicial complex, consider a *filtration*,

$$K^0 \subseteq K^1 \subseteq \cdots \subseteq K^N.$$

Then look at this evolution at the level of homology

$$H_d^0(K) \hookrightarrow H_d^1(K) \hookrightarrow \cdots \hookrightarrow H_d^N(K)$$

What is persistence?

Instead of a single simplicial complex, consider a *filtration*,

$$K^0 \subseteq K^1 \subseteq \cdots \subseteq K^N.$$

Then look at this evolution at the level of homology

$$H_d^0(K) \hookrightarrow H_d^1(K) \hookrightarrow \cdots \hookrightarrow H_d^N(K)$$

The steps of the filtration can be thought of as time

What is persistence?

Because of functoriality, there are maps between the different steps of homology,

$$f_i^j : H_d^i(K) \to H_d^j(K).$$

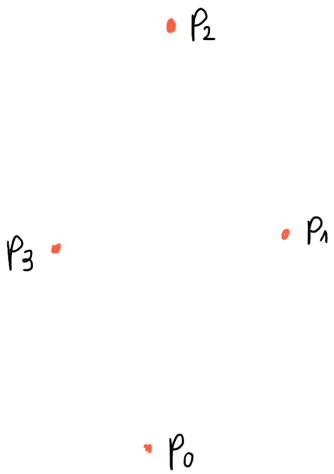
What is persistence?

Because of functoriality, there are maps between the different steps of homology,

$$f_i^j : H_d^i(K) \to H_d^j(K).$$

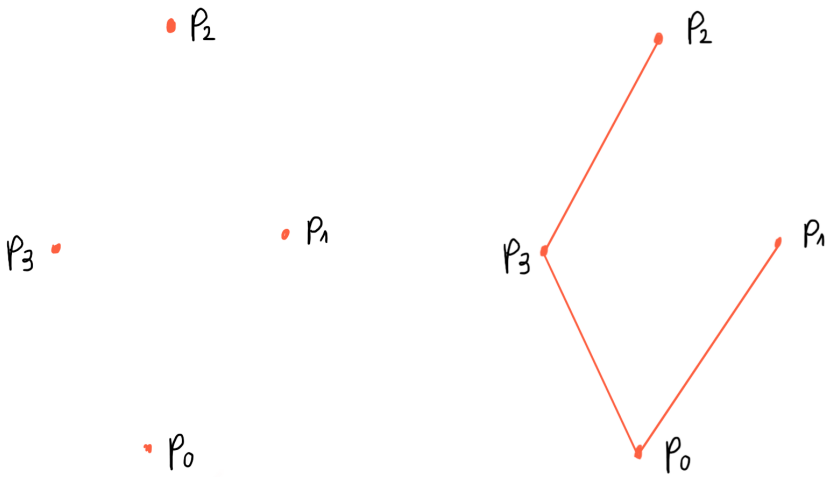
At each step, classes can be born (not in the image of f_i^{i+1}), die (in the kernel of f_i^{i+1}) or merge (have the same image through f_i^{i+1}).

An example



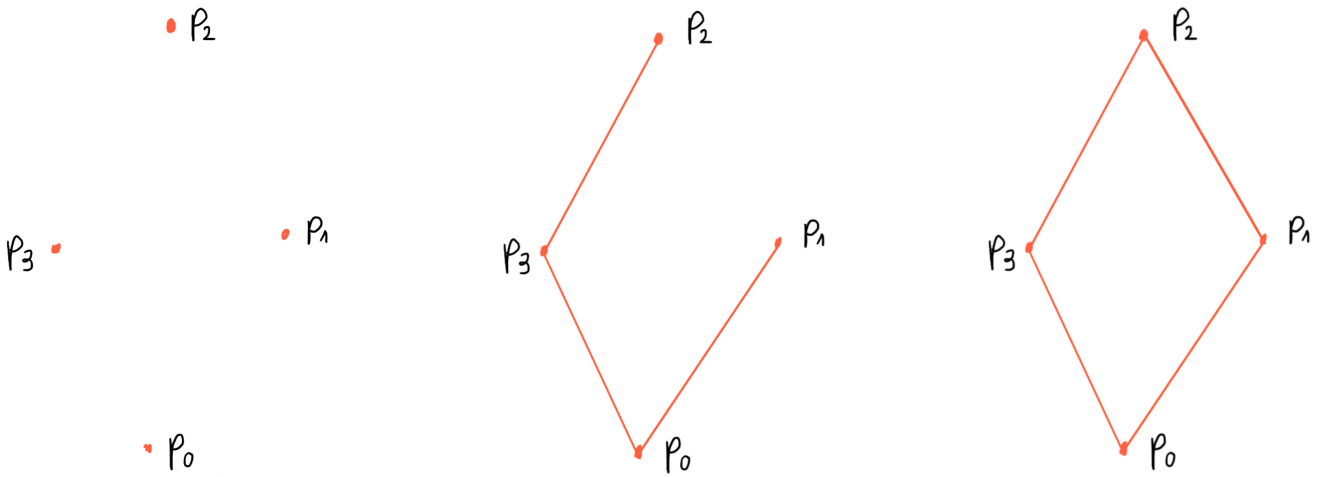
$[p_0], [p_1], [p_2], [p_3]$ are born.

An example



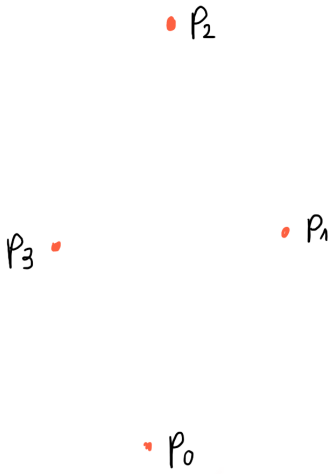
$[p_0], [p_1], [p_2], [p_3]$ merge.

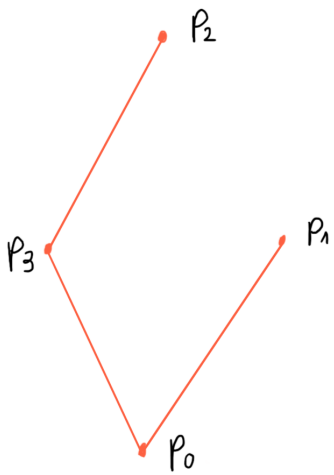
An example

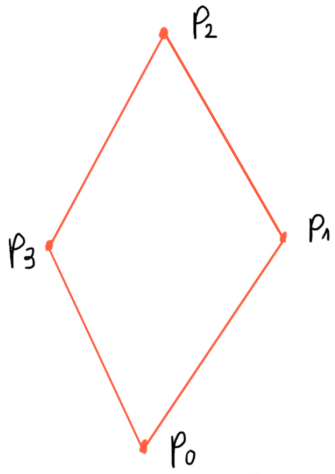


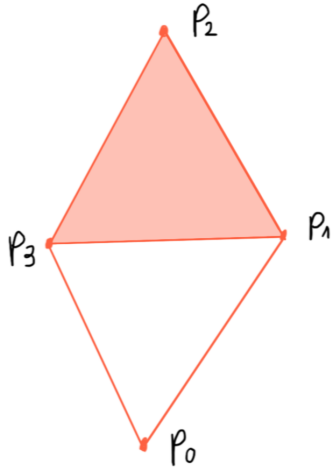
The cycle $[p_0, p_1] + [p_1, p_2] + [p_2, p_3] + [p_3, p_4]$ is born.

An example







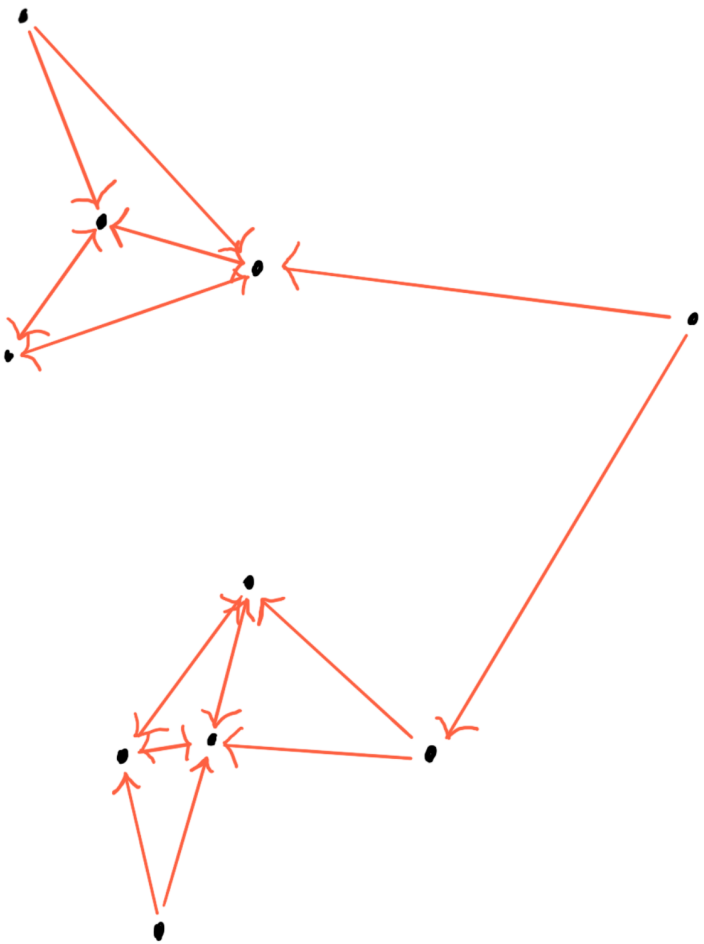


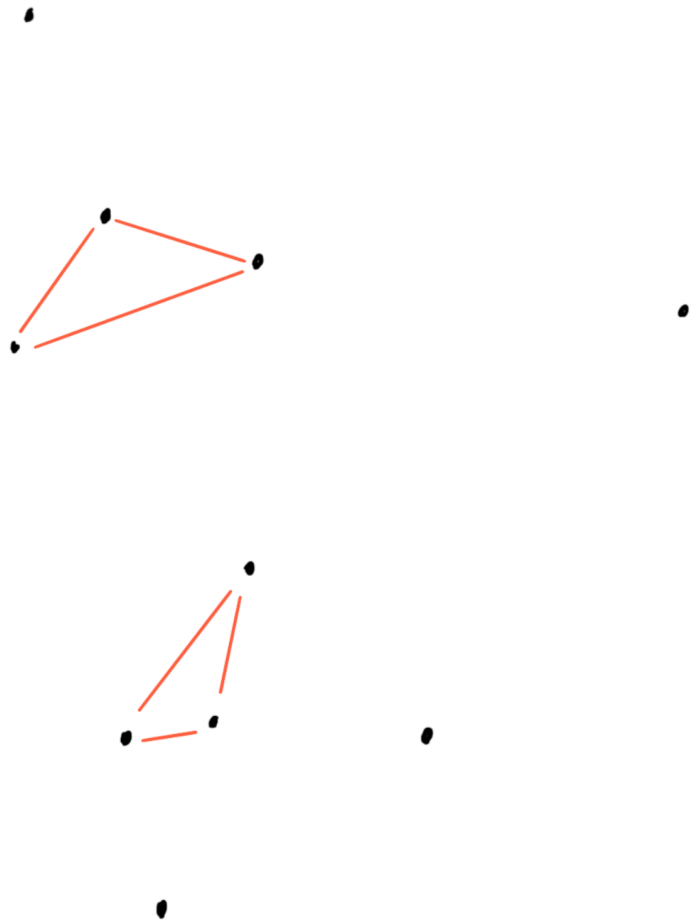
The cycle does not die.

Outline

Why the MkNN graph?

It is good for detecting clusters in a data set.





The outlier detection algorithm

The algorithm has four main parts:

The outlier detection algorithm

The algorithm has four main parts:

- The class `Cloud` stores a point cloud and computes its MkNN graph.

The outlier detection algorithm

The algorithm has four main parts:

- The class `Cloud` stores a point cloud and computes its MkNN graph.
- The class `Filtration` builds up the filtration.

The outlier detection algorithm

The algorithm has four main parts:

- The class `Cloud` stores a point cloud and computes its MkNN graph.
- The class `Filtration` builds up the filtration.
- The class `Homology` computes the homology.

The outlier detection algorithm

The algorithm has four main parts:

- The class `Cloud` stores a point cloud and computes its MkNN graph.
- The class `Filtration` builds up the filtration.
- The class `Homology` computes the homology.
- The persistence data of each class is gathered and plotted to detect the outliers.

The outlier detection algorithm

The algorithm has four main parts:

- The class `Cloud` stores a point cloud and computes its MkNN graph.
- The class `Filtration` builds up the filtration.
- The class `Homology` computes the homology.
- The persistence data of each class is gathered and plotted to detect the outliers.

All of the Python code is available at `github.com/arnaumas/mknn-homology`.

The class `Cloud`

- Store the cloud of N points in $\mathbb{R}^d$ as a NumPy array of shape (N, d).

The class `Cloud`

- Store the cloud of N points in $\mathbb{R}^d$ as a NumPy array of shape (N, d).
- Construct the adjacency matrix, A, of the k-nearest neighbours graph of the cloud.

The class `Cloud`

- Store the cloud of N points in $\mathbb{R}^d$ as a NumPy array of shape (N, d).
- Construct the adjacency matrix, A, of the k-nearest neighbours graph of the cloud.
- The adjacency matrix of the the MkNN graph is

$$M = A \odot A^\top$$

where $\odot$ is the entrywise product.

The class `Filtration`

- The package NetworkX can compute the cliques of a graph. These correspond to the simplices of the *graph complex* associated to the MkNN graph of the cloud.

The class `Filtration`

- The package NetworkX can compute the cliques of a graph. These correspond to the simplices of the *graph complex* associated to the MkNN graph of the cloud.
- The simplices are ordered by order of appearance and dimension, which ensures any simplex is always preceded by its faces.

The class `Homology`

- The classes `Simplex` and `Chain` model elements of the chain groups and underpin the following computations.

The class `Homology`

- The classes `Simplex` and `Chain` model elements of the chain groups and underpin the following computations.
- The chain group is taken over $\mathbb{F}_2$, which corresponds to disregarding orientation and simplifies computation.

The class `Homology`

- The classes `Simplex` and `Chain` model elements of the chain groups and underpin the following computations.
- The chain group is taken over $\mathbb{F}_2$, which corresponds to disregarding orientation and simplifies computation.
- There is a Python dictionary which keeps track of the homology class of every clique. This is a memoised version of the projection $\pi\colon C_n(K) \to C_n(K)/B_n(K)$.

The class `Homology`

- The classes `Simplex` and `Chain` model elements of the chain groups and underpin the following computations.
- The chain group is taken over $\mathbb{F}_2$, which corresponds to disregarding orientation and simplifies computation.
- There is a Python dictionary which keeps track of the homology class of every clique. This is a memoised version of the projection $\pi\colon C_n(K) \to C_n(K)/B_n(K)$.

The class `Homology`

- Iterating through the filtration, for every simplex σ, the homology class of the sum of its faces, $w_0 + \cdots + w_n$.

- Iterating through the filtration, for every simplex σ, the homology class of the sum of its faces, $w_0 + \cdots + w_n$.
- If it is zero, then before σ appears, there is chain c such that

$$w_0 + \cdots + w_n = \partial c.$$

- Iterating through the filtration, for every simplex σ, the homology class of the sum of its faces, $w_0 + \cdots + w_n$.
- If it is zero, then before σ appears, there is chain c such that

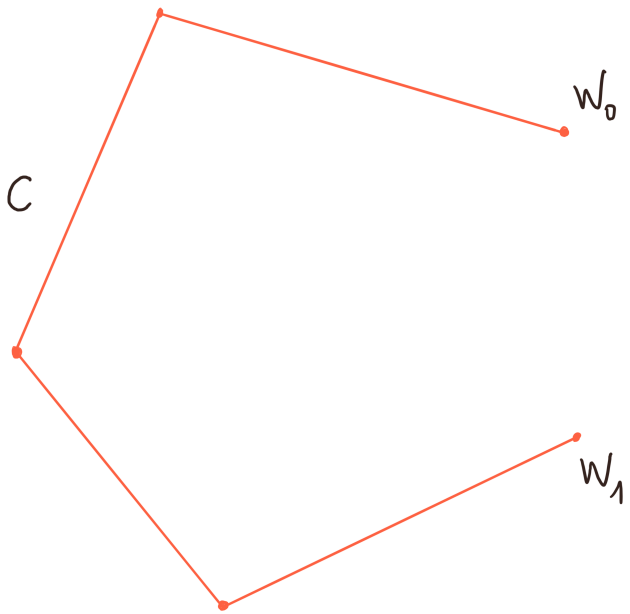
$$w_0 + \cdots + w_n = \partial c.$$

Once σ is added,

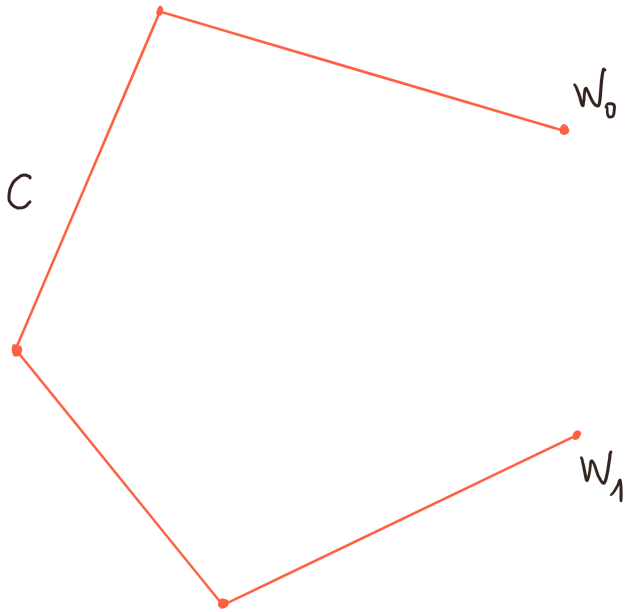
$$w_0 + \cdots + w_n = \partial c = \partial \sigma$$

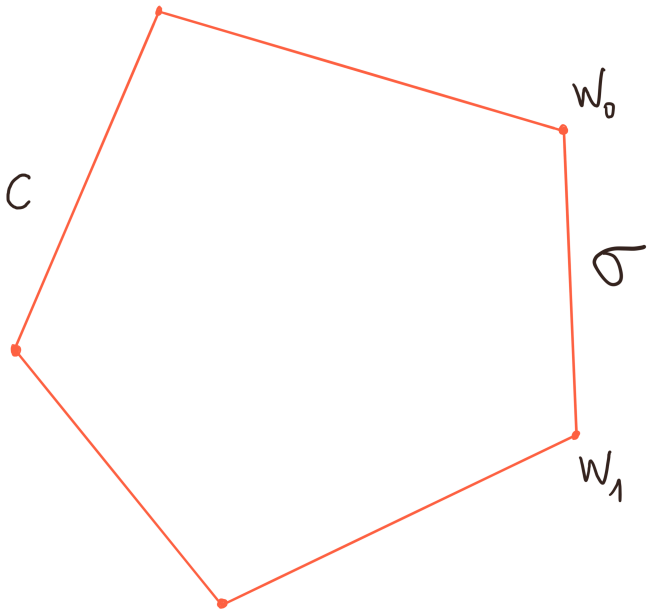
so $\partial(\sigma + c) = 0$, which means $\sigma + c$ is a new cycle.

The class Homology



The class Homology





The class `Homology`

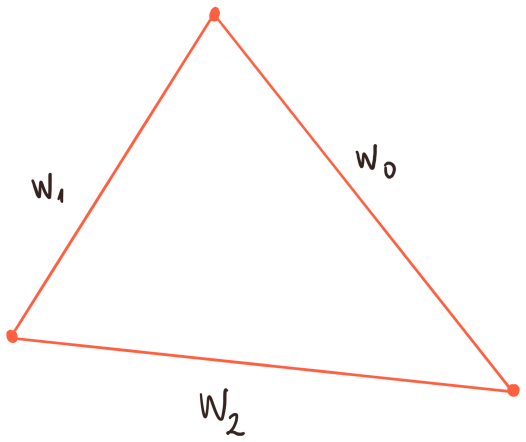
- If it is not zero, then, before σ appears, its faces formed a cycle not homologous to zero. Therefore its homology class was a nontrivial element of the corresponding homology group.

The class `Homology`

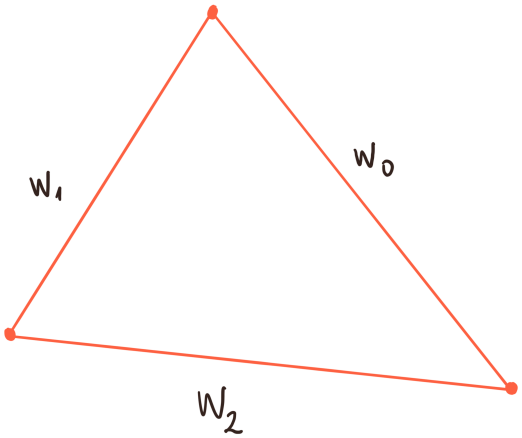
- If it is not zero, then, before σ appears, its faces formed a cycle not homologous to zero. Therefore its homology class was a nontrivial element of the corresponding homology group.
- Once σ appears, it is covered and so it dies.

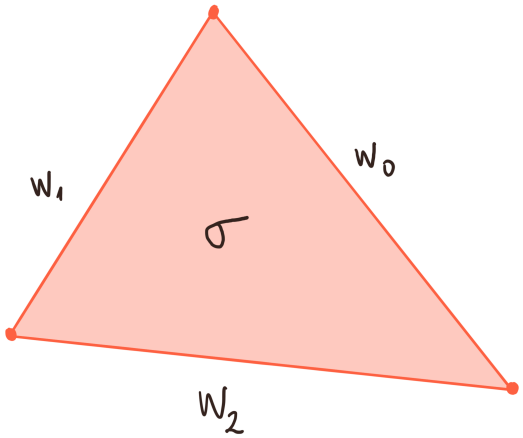
The class Homology

- If it is not zero, then, before σ appears, its faces formed a cycle not homologous to zero. Therefore its homology class was a nontrivial element of the corresponding homology group.
- Once σ appears, it is covered and so it dies.



- If it is not zero, then, before σ appears, its faces formed a cycle not homologous to zero. Therefore its homology class was a nontrivial element of the corresponding homology group.
- Once σ appears, it is covered and so it dies.





The persistence data

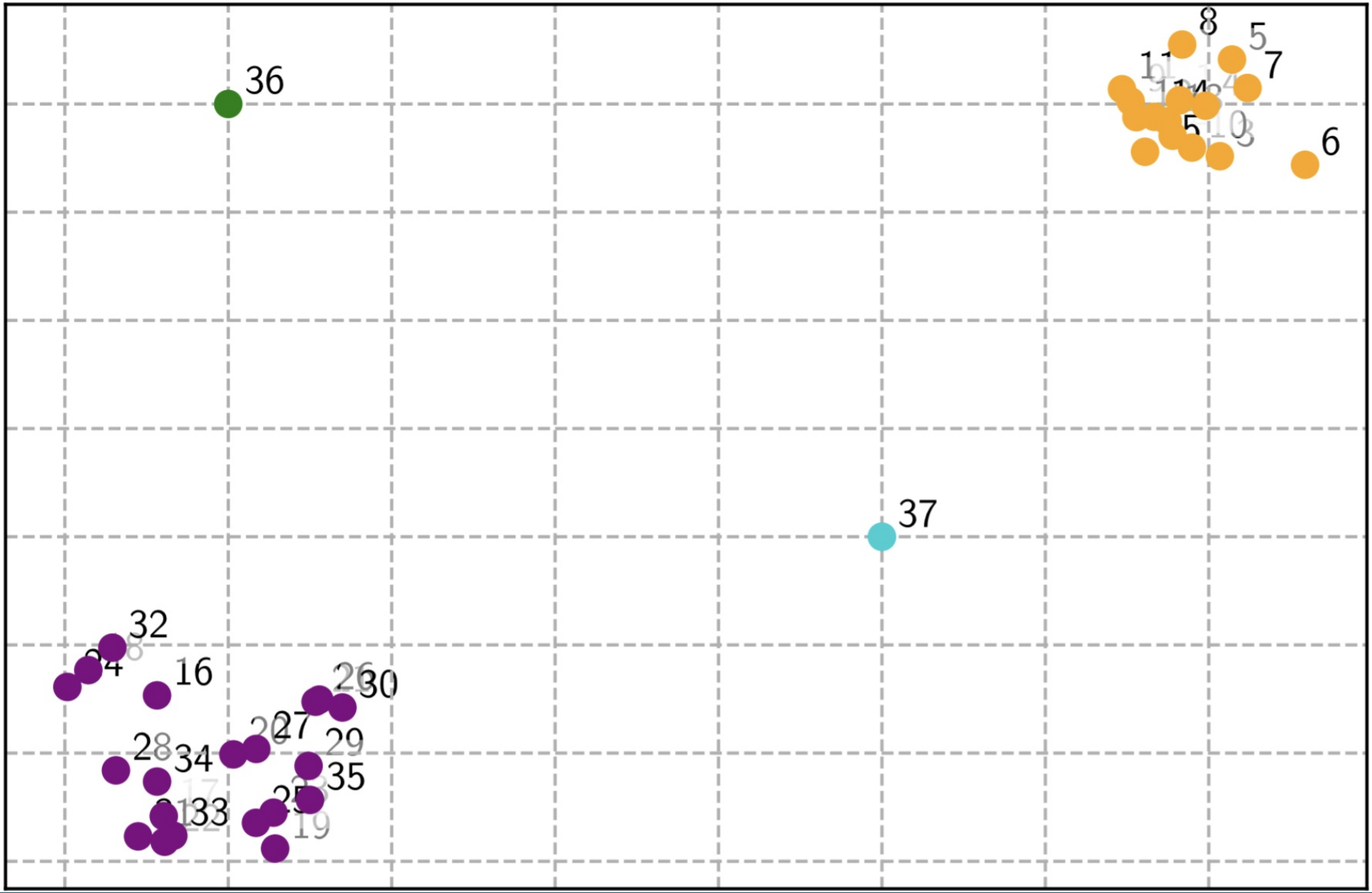
The lifetime and size at death of every class in the zeroth homology group is stored and plotted. Outliers will be those classes with few members and long lifetimes, since they took a long time to merge into larger clusters.

The persistence data

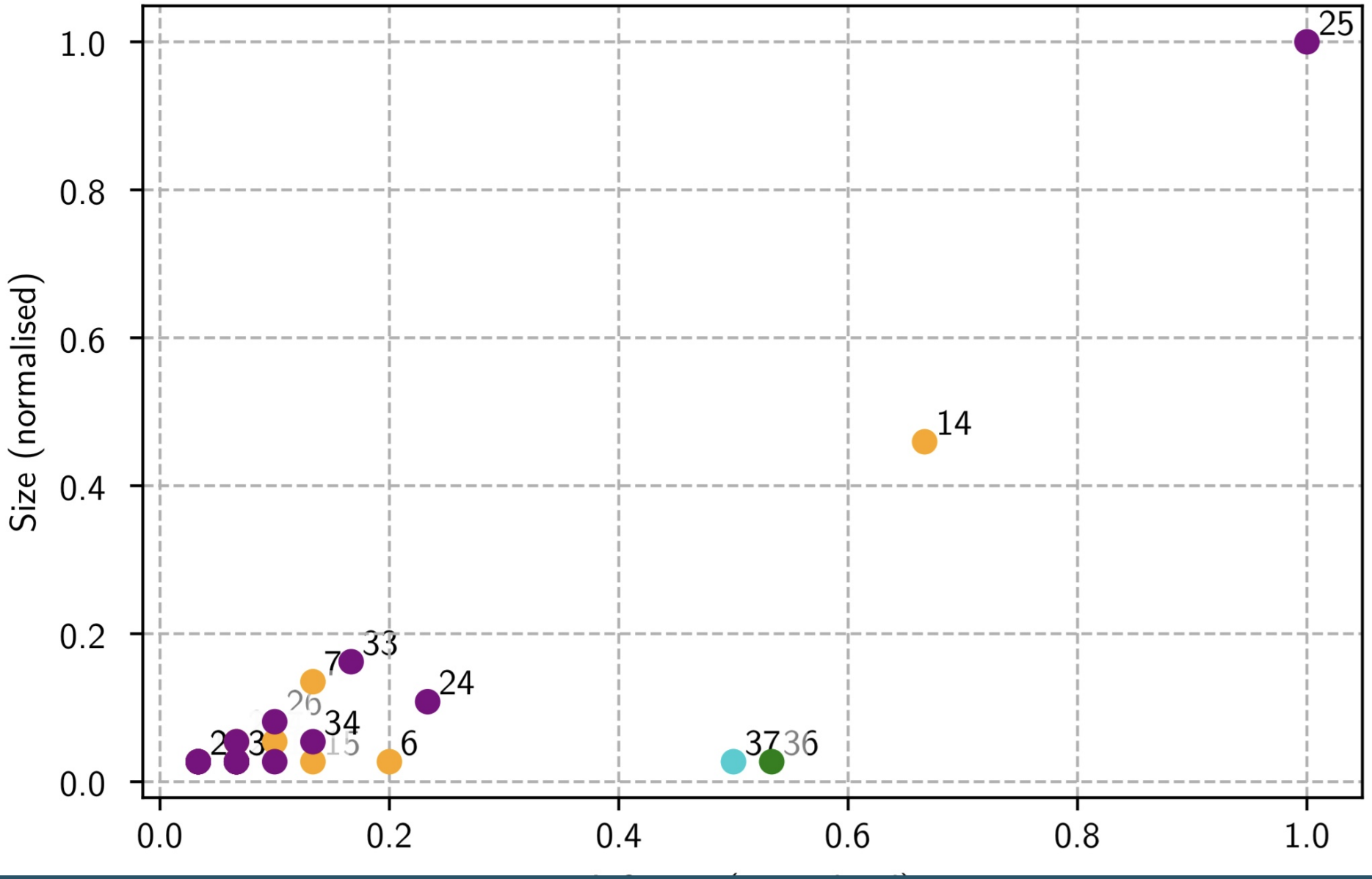
The lifetime and size at death of every class in the zeroth homology group is stored and plotted. Outliers will be those classes with few members and long lifetimes, since they took a long time to merge into larger clusters. Visually, outliers will be those points in the lower right corner of the plot.

Outline

A run with toy data



A run with toy data



A run with real data

- The detector was run with a data set consisting of 26 radiomic features measured from 20 different patients. These are the result of a previous study carried out at VHIO.

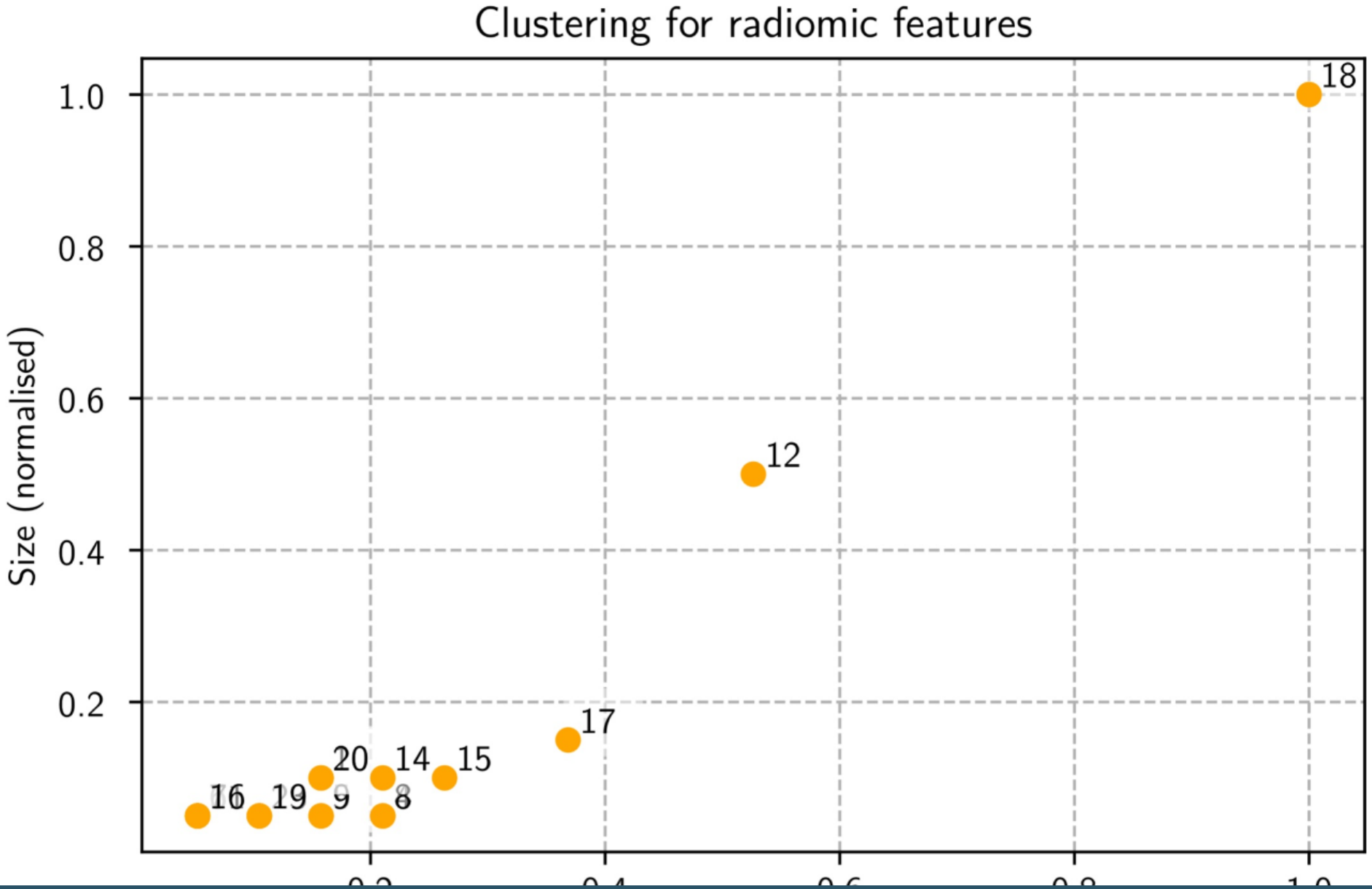
A run with real data

- The detector was run with a data set consisting of 26 radiomic features measured from 20 different patients. These are the result of a previous study carried out at VHIO.
- There are also genomic features corresponding to the same 20 patients.

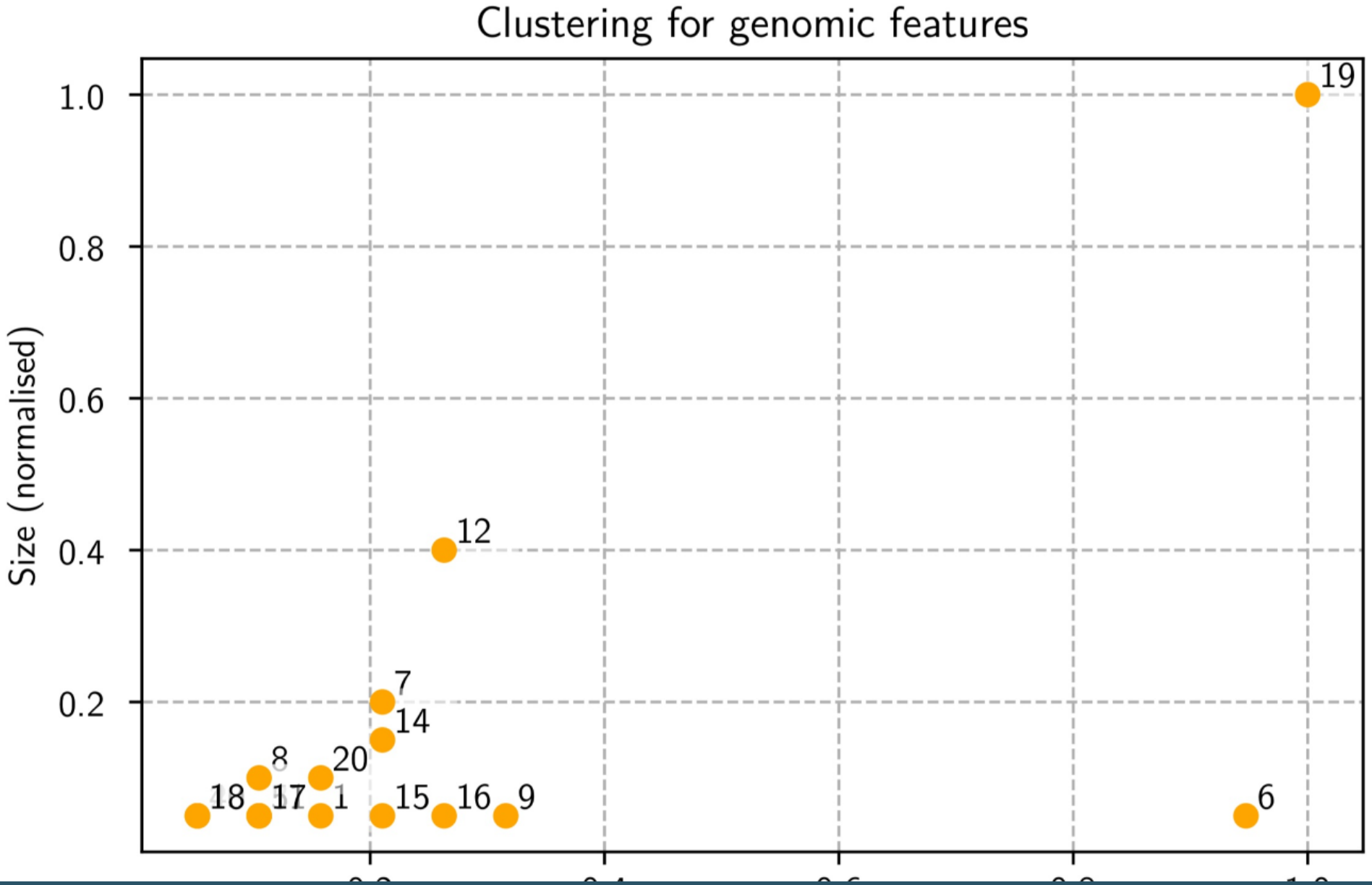
A run with real data

- The detector was run with a data set consisting of 26 radiomic features measured from 20 different patients. These are the result of a previous study carried out at VHIO.
- There are also genomic features corresponding to the same 20 patients.
- The detector was run on both sets to compare the clustering structure.

A run with real data



A run with real data



Conclusions

- Detector appears to properly detect outliers, and gives a quantitative measure of “outlierness”.

Conclusions

- Detector appears to properly detect outliers, and gives a quantitative measure of “outlierness”.
- The results from the real dataset are awaiting an assesment from VHIO on their medical significance.

Conclusions

- Detector appears to properly detect outliers, and gives a quantitative measure of “outlierness”.
- The results from the real dataset are awaiting an assesment from VHIO on their medical significance.
- Further considerations: efficiency, edgecase testing, higher dimensional homology.

Topological data analysis and persistent homology

Developing an outlier detector based on persistent homology

Arnau Mas

Supervised by
Dr Albert Ruiz

Universitat Autònoma de Barcelona
September 3rd 2020

Universitat Autònoma
de Barcelona