# PersLay: a Neural Network for persistence diagrams
## and related topics.

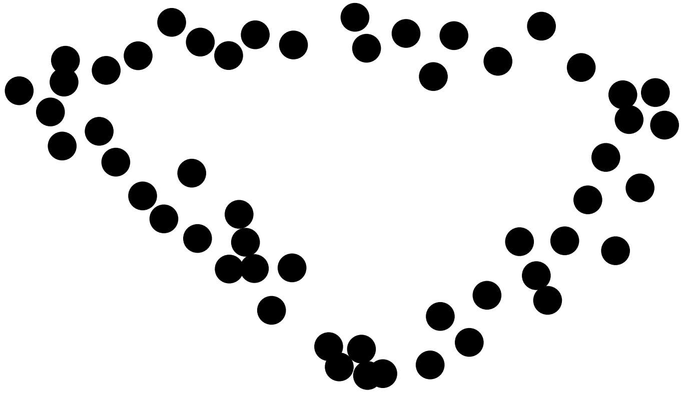
M. Carrière, F. Chazal, Y. Ike, **T. Lacombe**,
M. Royer, Y. Umeda

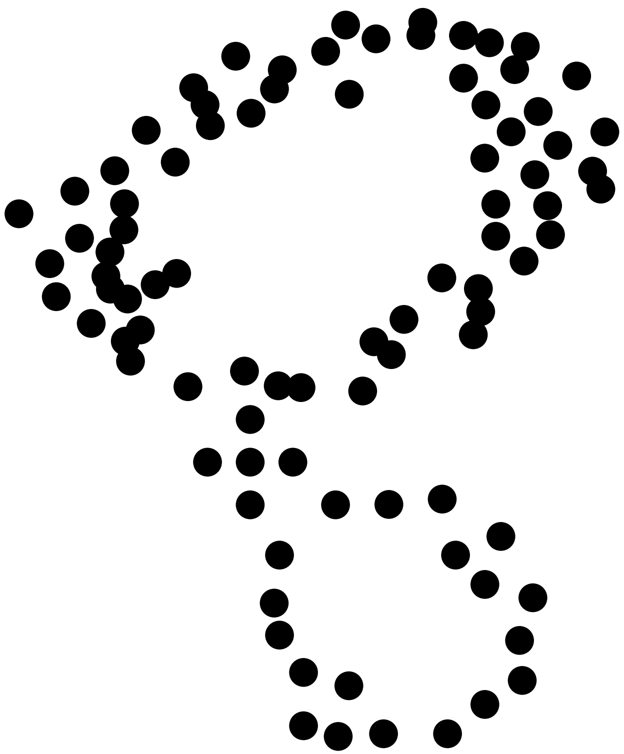
6th SmartData@Polito Workshop
Higher Methods in Data Science and ML

Jan. 30th, 2020.

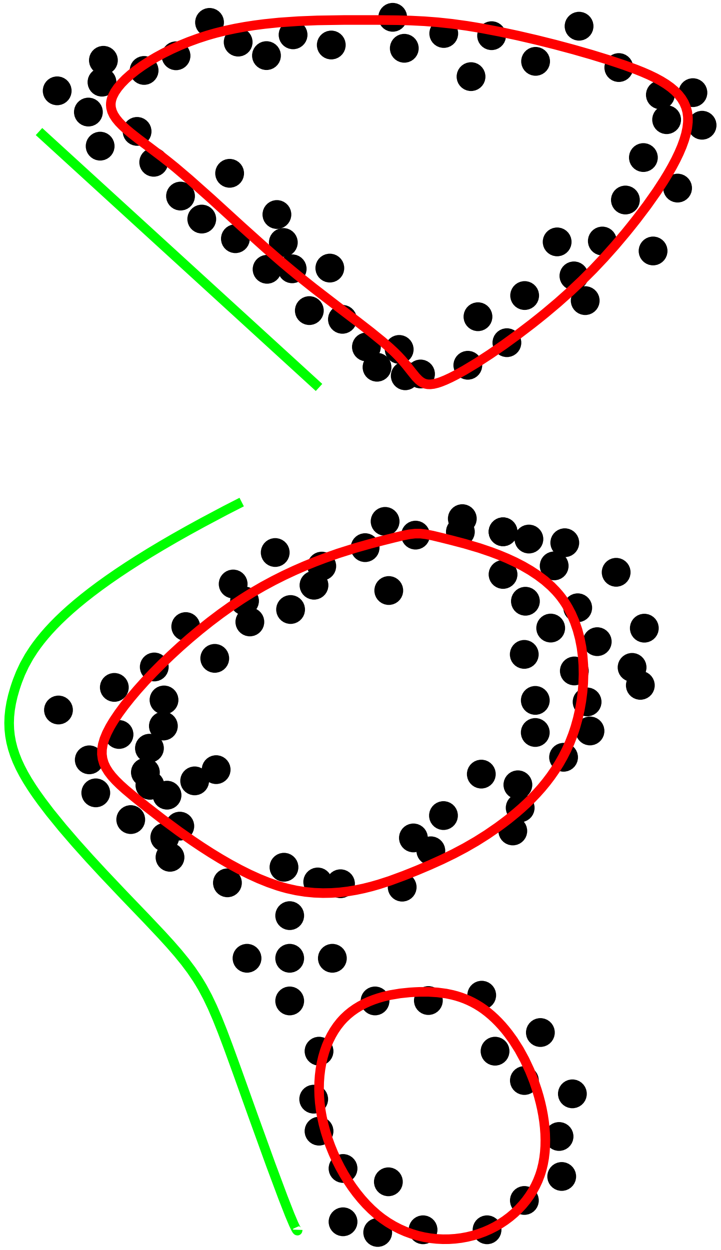
DataShape - Inria Saclay
theo.lacombe@inria.fr

# Preliminar ideas about TDA

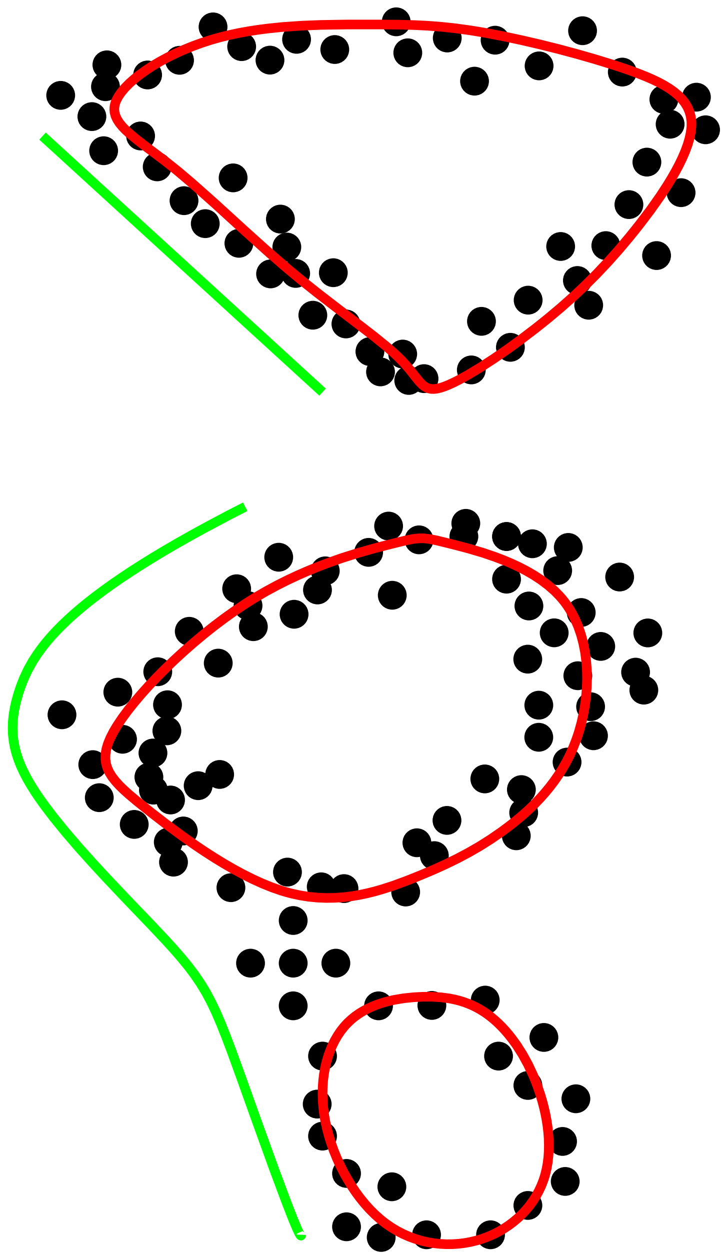




# Preliminar ideas about TDA



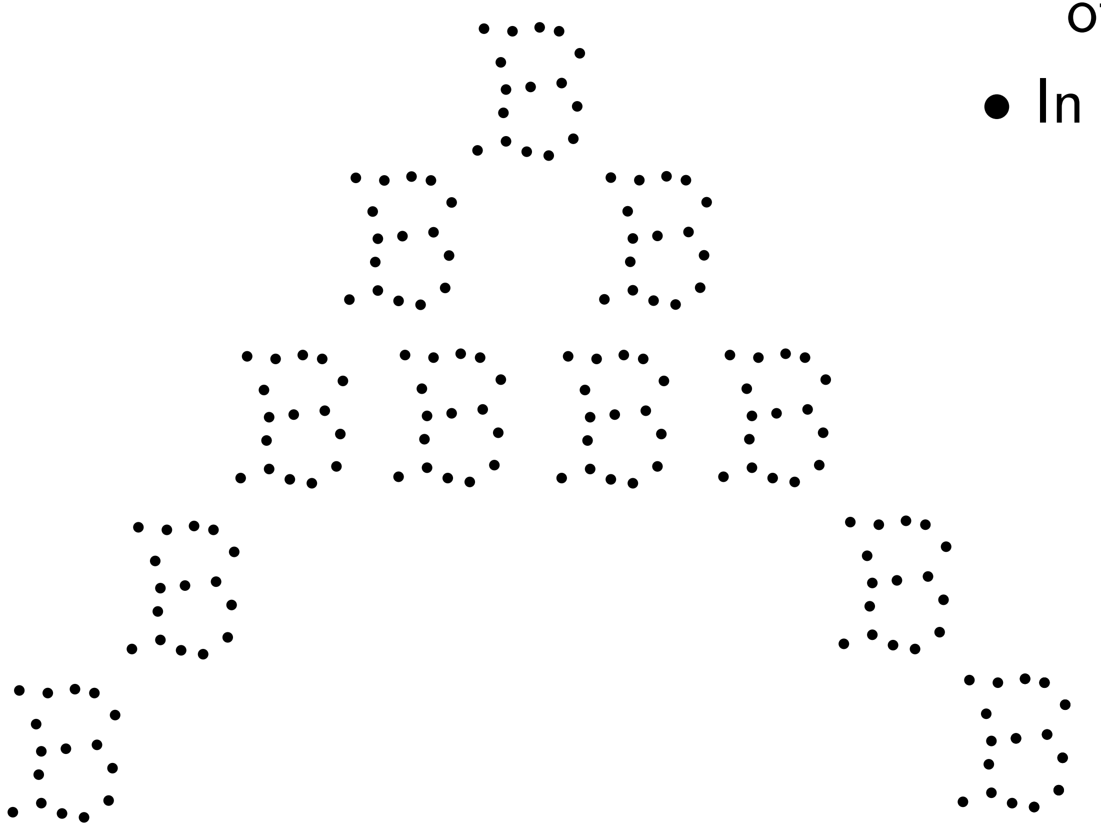
# Preliminar ideas about TDA



## TDA is about:

- Providing topological **descriptors** of a given object
- In a quantitative way.

# Preliminar ideas about TDA



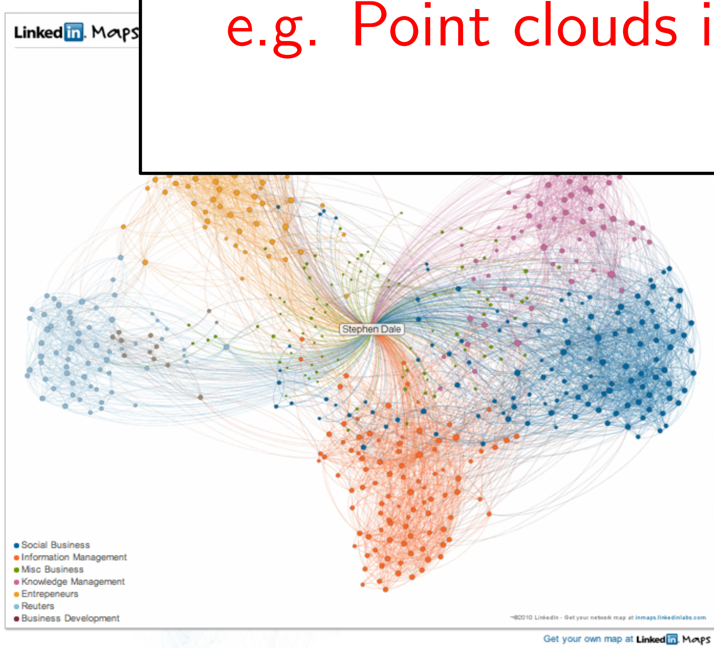
## TDA is about:

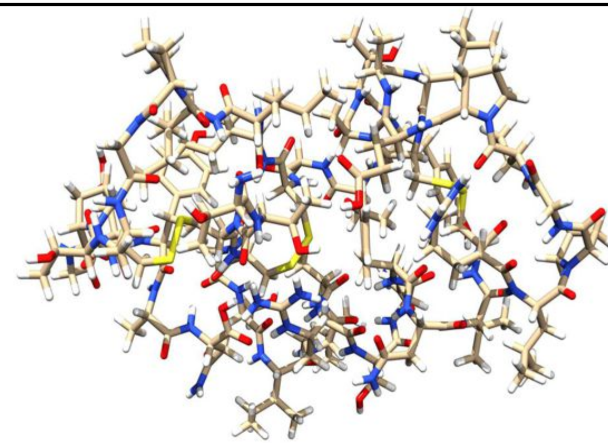
- Providing topological **descriptors** of a given object
- In a quantitative way.

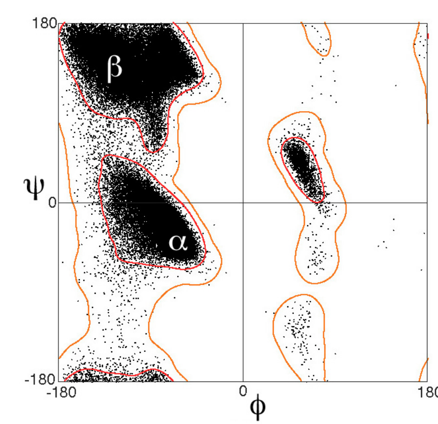
# A global overview of TDA


TDA can be applied to any type of structured data.

e.g. Point clouds in $\mathbb{R}^d$, Weighted graphs, time series...





# The TDA pipeline: persistent homology

## Reminder:

We want to encode the topology of an object at all scales

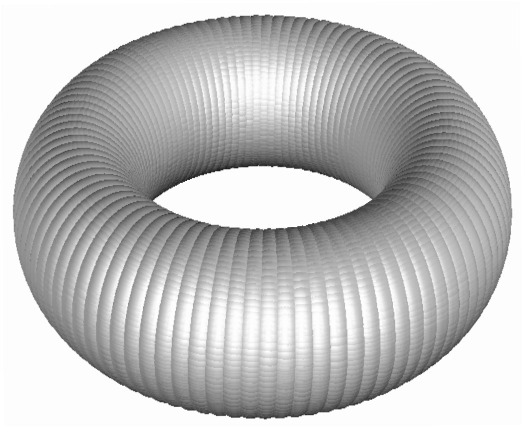
# The TDA pipeline: persistent homology

## Reminder:

We want to encode the topology of an object at all scales

## Homology:

Encodes the topological features (connected components, loop, holes...)



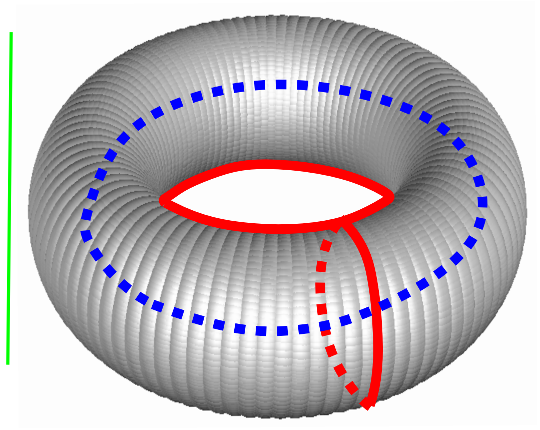
# The TDA pipeline: persistent homology

## Reminder:

We want to encode the topology of an object at all scales

## Homology:

Encodes the topological features (connected components, loop, holes...)



$\beta_0 = 1$ connected component
$\beta_1 = 2$ loops
$\beta_2 = 1$ hole

# The TDA pipeline: persistent homology

## Reminder:

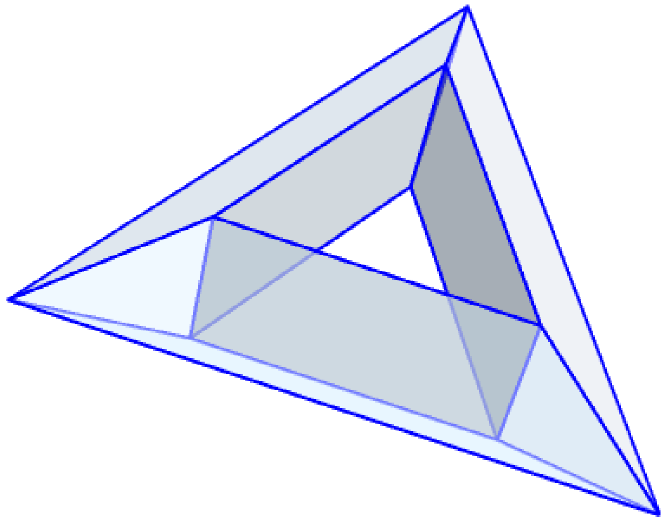
We want to encode the topology of an object at all scales

## Homology:

Encodes the topological features (connected components, loop, holes...)

## Simplicial homology:

Homology on a computer.



$\beta_0 = 1$ connected component

$\beta_1 = 2$ loops

$\beta_2 = 1$ hole

# The TDA pipeline: persistent homology

## Reminder:

We want to encode the topology of an object at all scales

## Persistence: looking at scales

All we need is an increasing sequence of topological spaces.

Example: Given a function $f : \mathbb{X} \to \mathbb{R}$, consider $F_t := f^{-1}((-\infty, t])$

filtration

$$\boxed{x \in F_t \Leftrightarrow f(x) \leq t}$$

sublevel sets of $f$

# The TDA pipeline: persistent homology

## Reminder:

We want to encode the topology of an object at all scales

## Persistence: looking at scales

All we need is an increasing sequence of topological spaces.

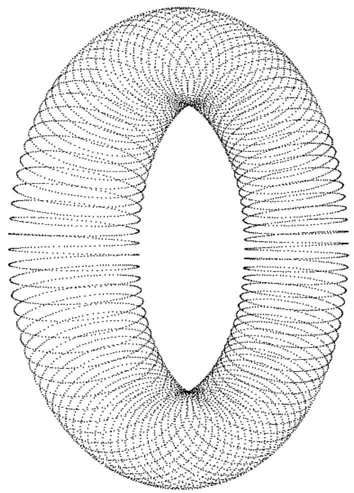
Example: Given a function $f : \mathbb{X} \to \mathbb{R}$, consider $F_t := f^{-1}((-\infty, t])$

filtration

$$\boxed{x \in F_t \Leftrightarrow f(x) \leq t}$$

sublevel sets of $f$

Example: Given $X = x_1 \ldots x_n \in \mathbb{R}^d =: \mathbb{X}$, take $f(x) = \mathrm{dist}_X(x)$



# The TDA pipeline: persistent homology

## Reminder:

We want to encode the topology of an object at all scales

## Persistence: looking at scales

All we need is an increasing sequence of topological spaces.

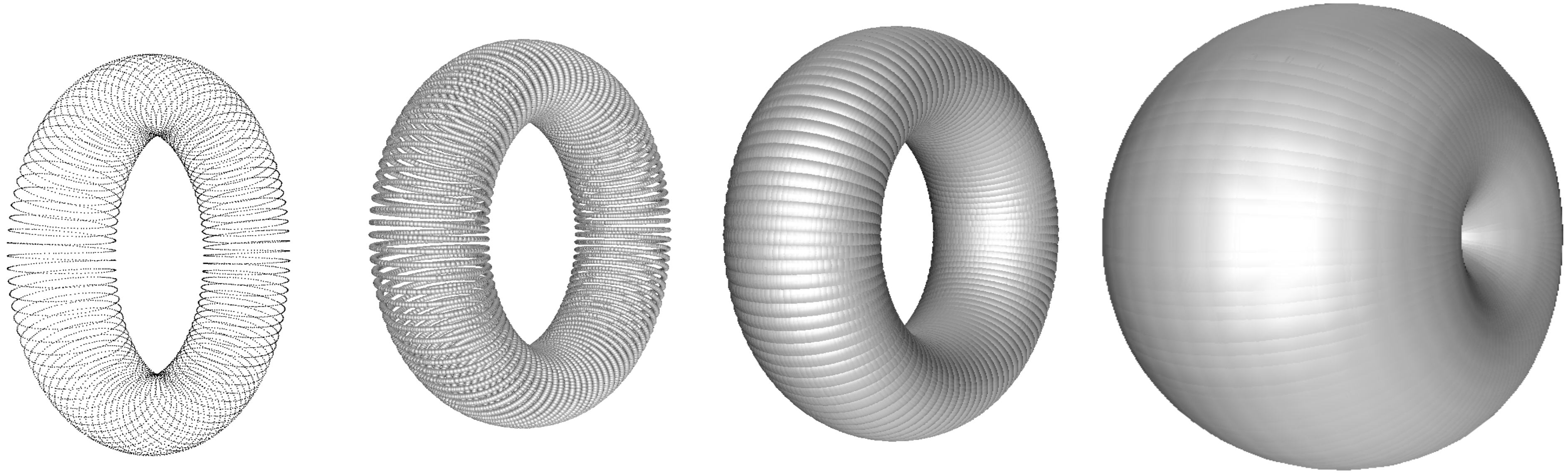
Example: Given a function $f : \mathbb{X} \to \mathbb{R}$, consider $F_t := f^{-1}((-\infty, t])$

filtration

$$\boxed{x \in F_t \Leftrightarrow f(x) \leq t}$$

sublevel sets of $f$

Example: Given $X = x_1 \dots x_n \in \mathbb{R}^d =: \mathbb{X}$, take $f(x) = \mathrm{dist}_X(x)$



# The TDA pipeline: persistent homology

- Compute *at all scale* the homology of the current topological space
- Record *births* and *deaths* of topological features (loop, holes, etc.)
- Print the output as the persistence diagram of the filtration

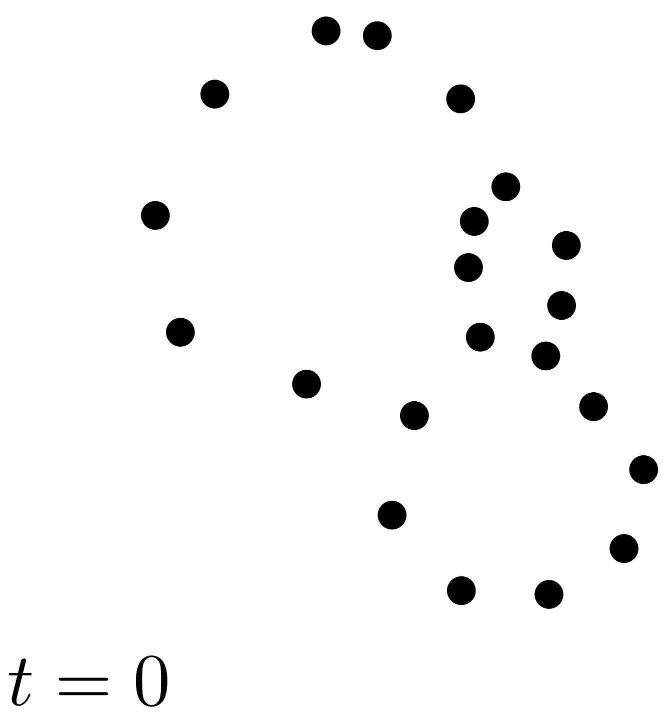
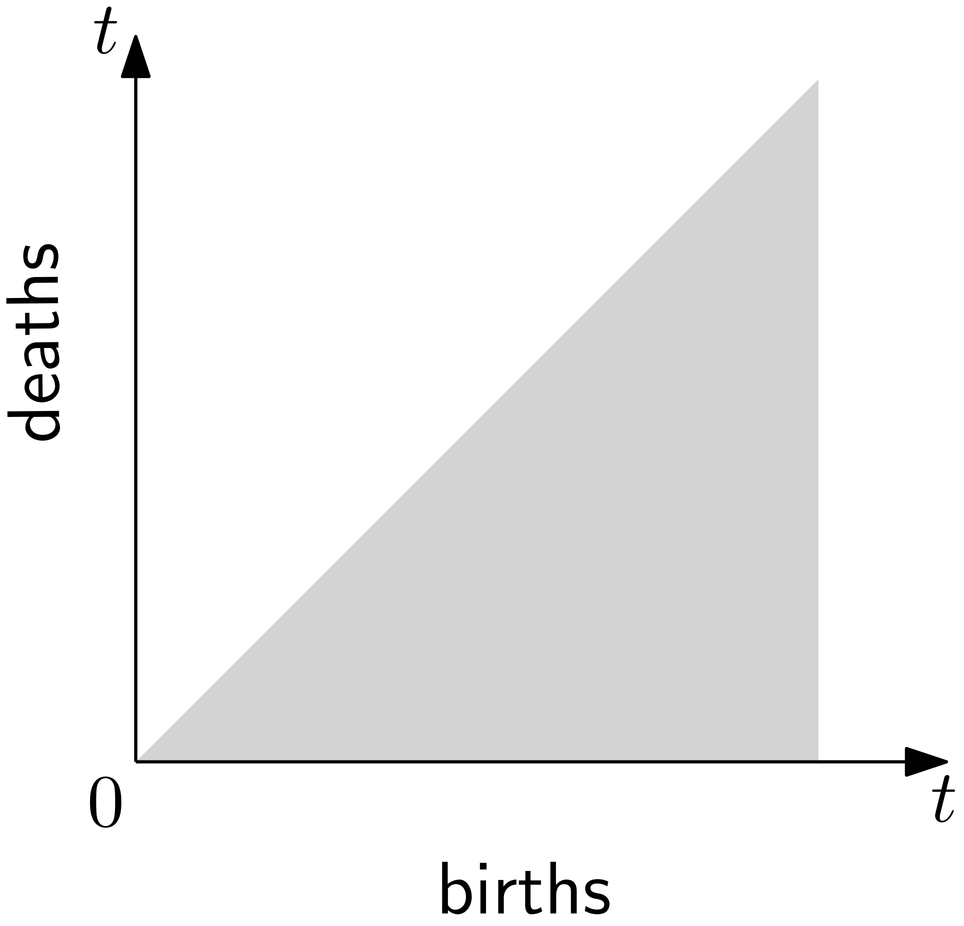



Diagram in dim $1$ (loops)



# The TDA pipeline: persistent homology

- Compute *at all scale* the homology of the current topological space
- Record *births* and *deaths* of topological features (loop, holes, etc.)
- Print the output as the persistence diagram of the filtration

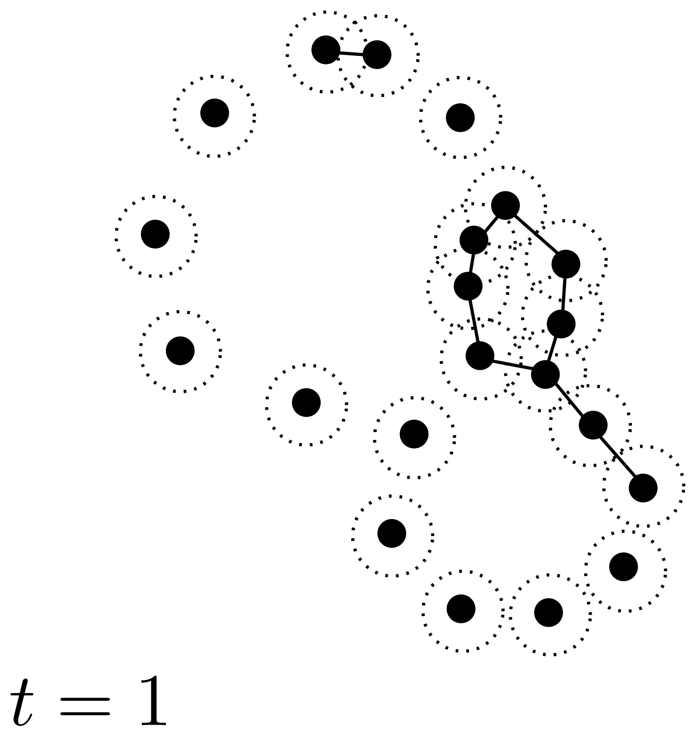
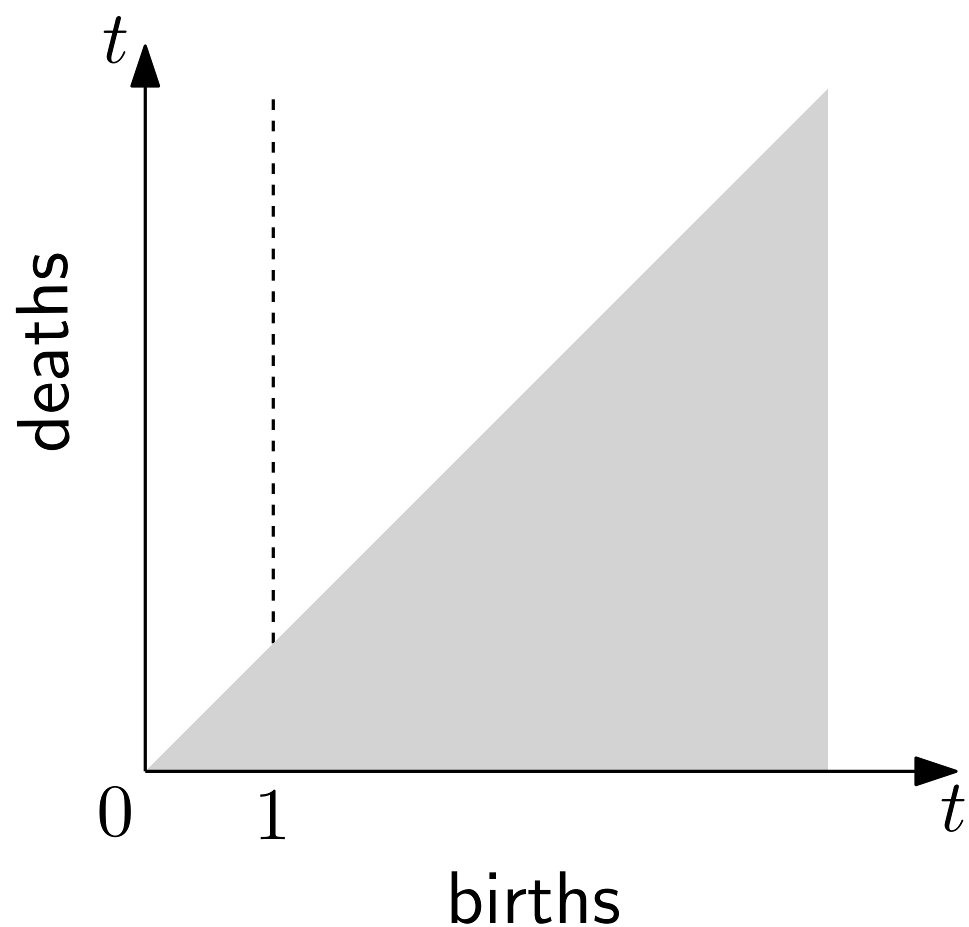
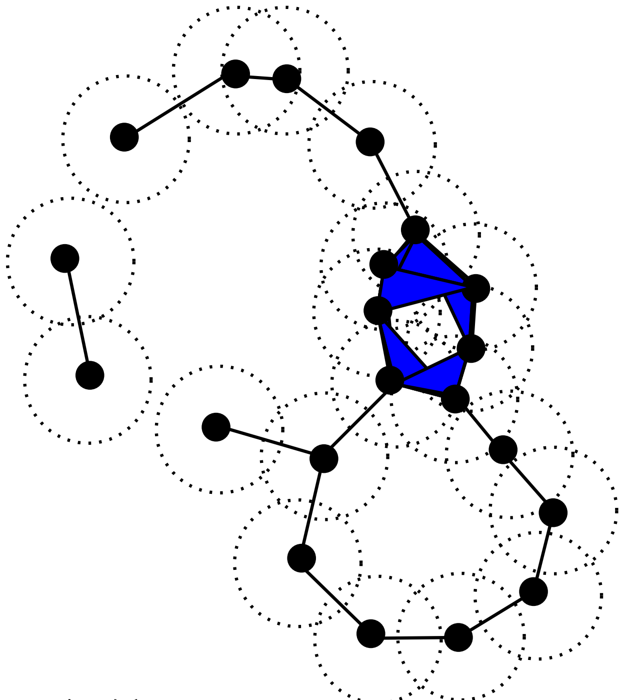



Diagram in dim $1$ (loops)



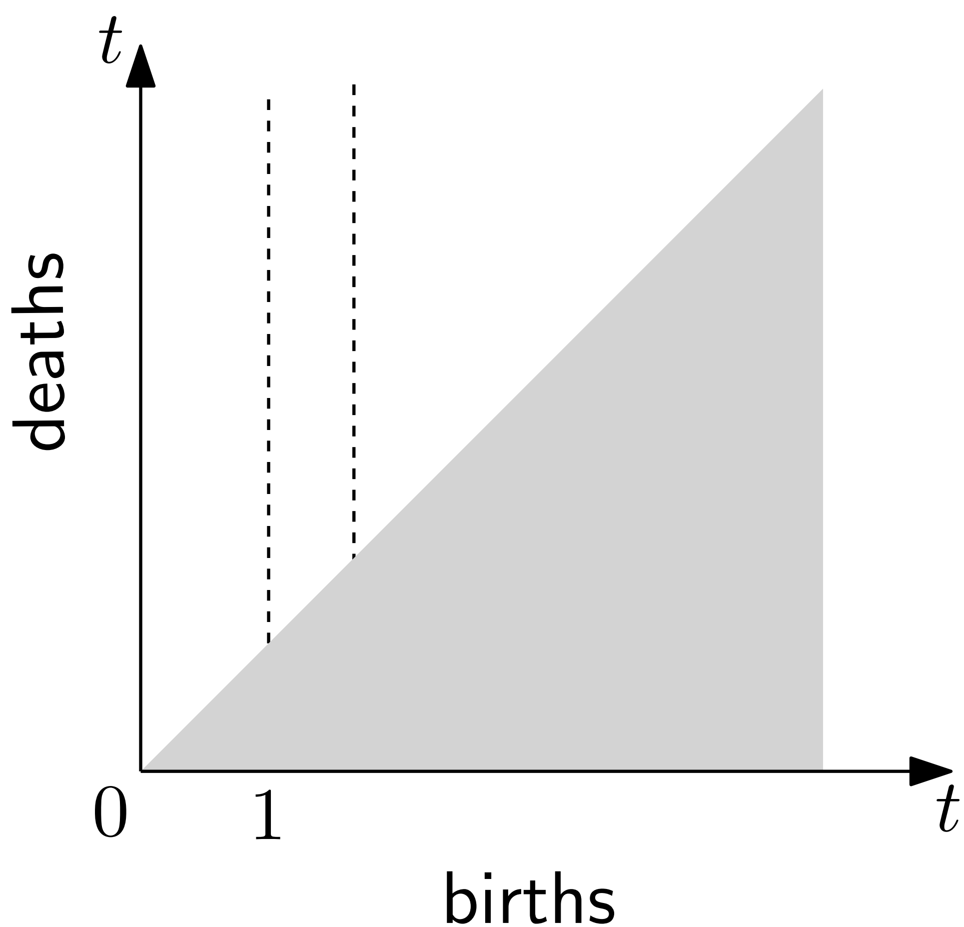
# The TDA pipeline: persistent homology

- Compute *at all scale* the homology of the current topological space
- Record *births* and *deaths* of topological features (loop, holes, etc.)
- Print the output as the persistence diagram of the filtration



$t = 1.5$

Diagram in dim $1$ (loops)



# The TDA pipeline: persistent homology

- Compute *at all scale* the homology of the current topological space
- Record *births* and *deaths* of topological features (loop, holes, etc.)
- Print the output as the persistence diagram of the filtration

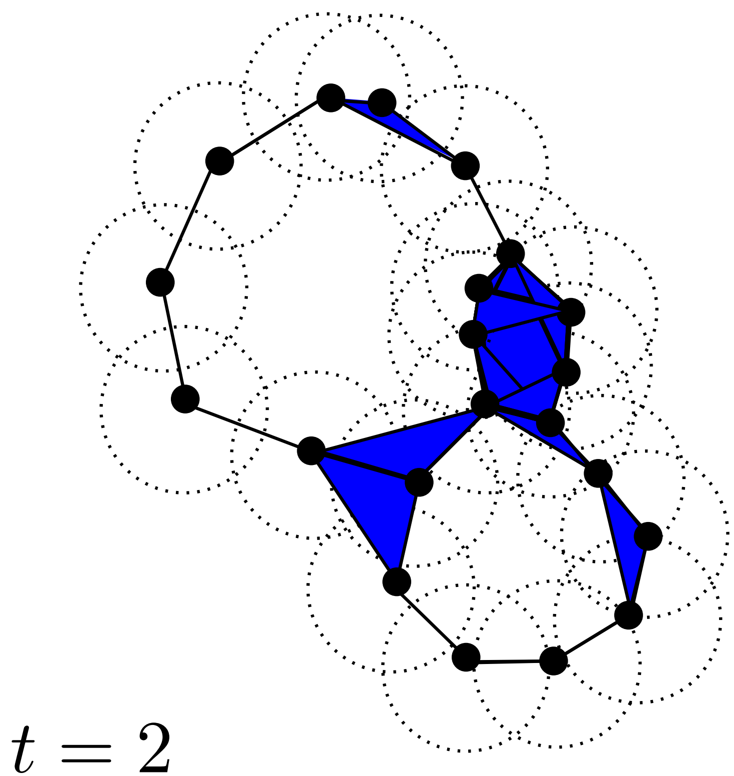
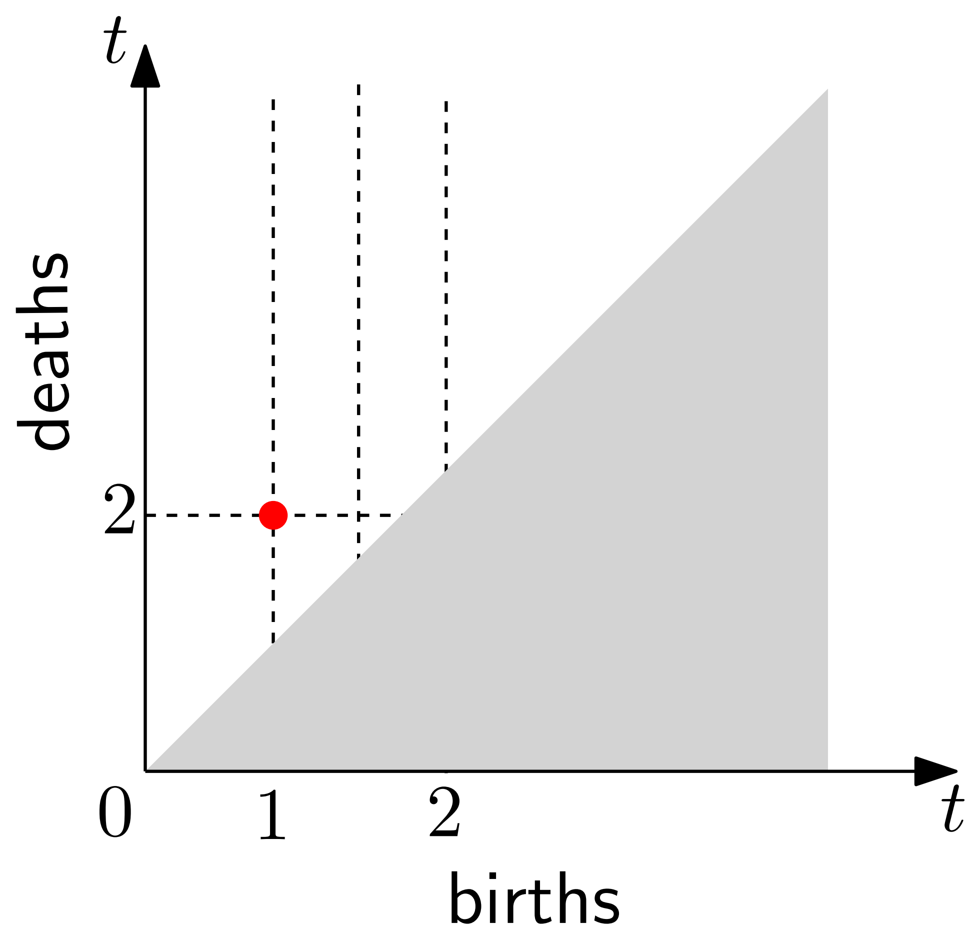



Diagram in dim $1$ (loops)



# The TDA pipeline: persistent homology

- Compute *at all scale* the homology of the current topological space
- Record *births* and *deaths* of topological features (loop, holes, etc.)
- Print the output as the persistence diagram of the filtration

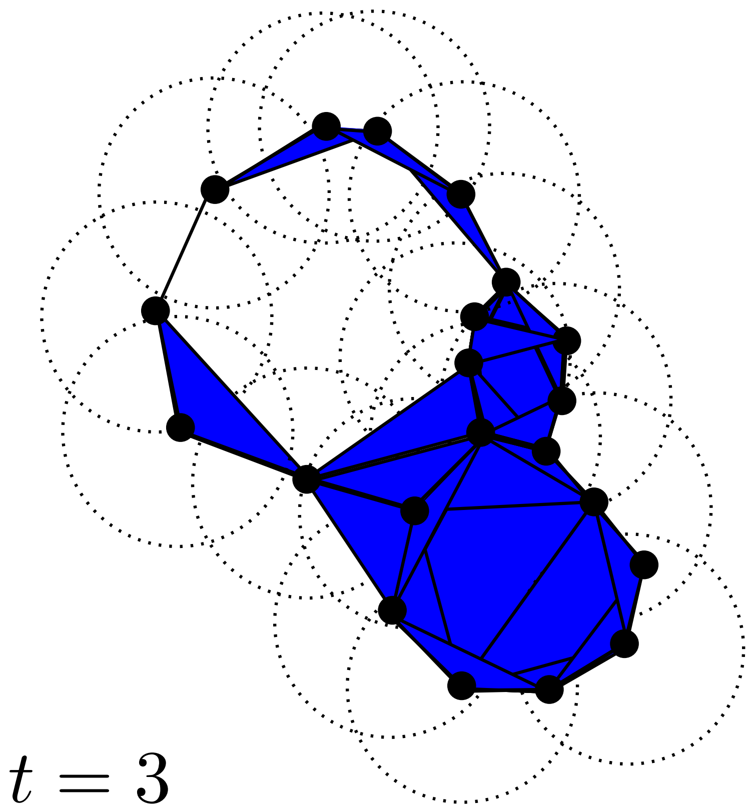
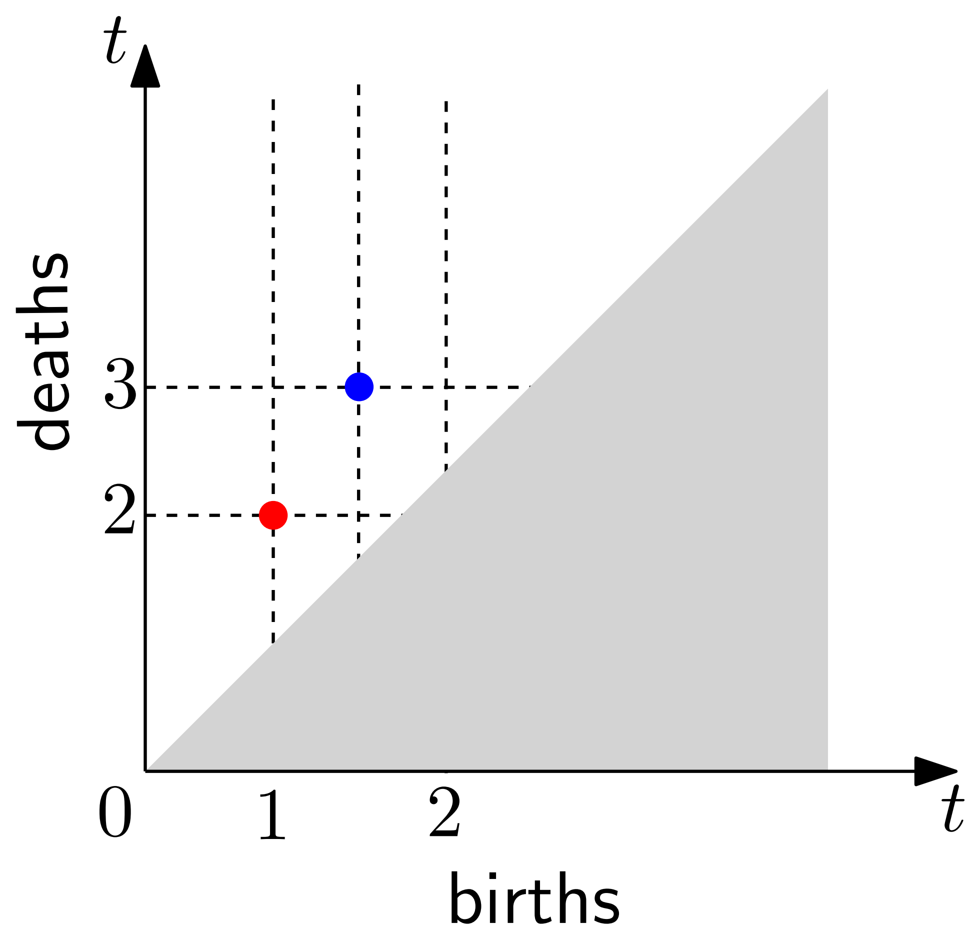
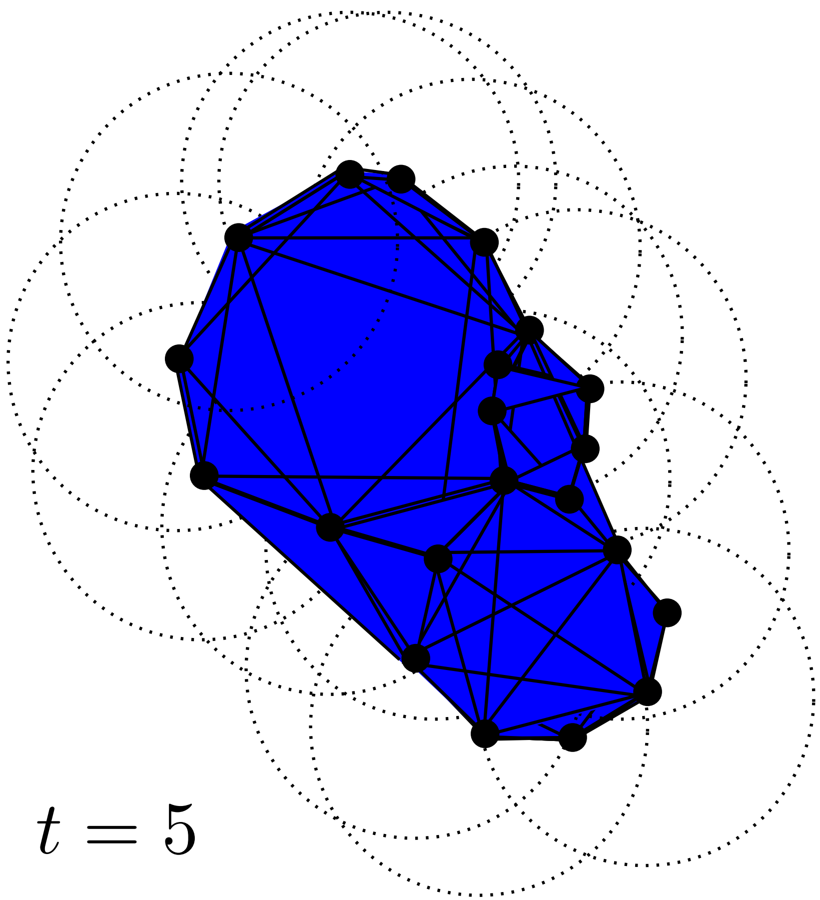


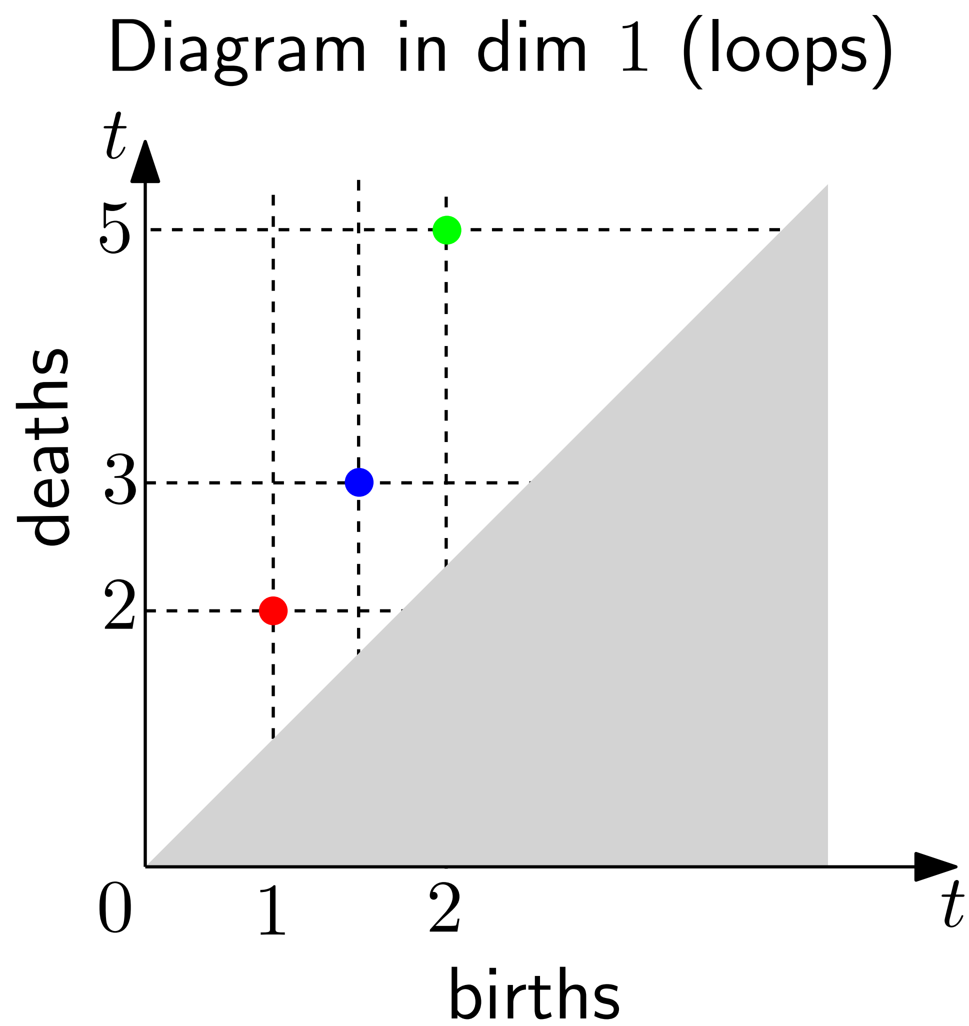

Diagram in dim $1$ (loops)



# The TDA pipeline: persistent homology

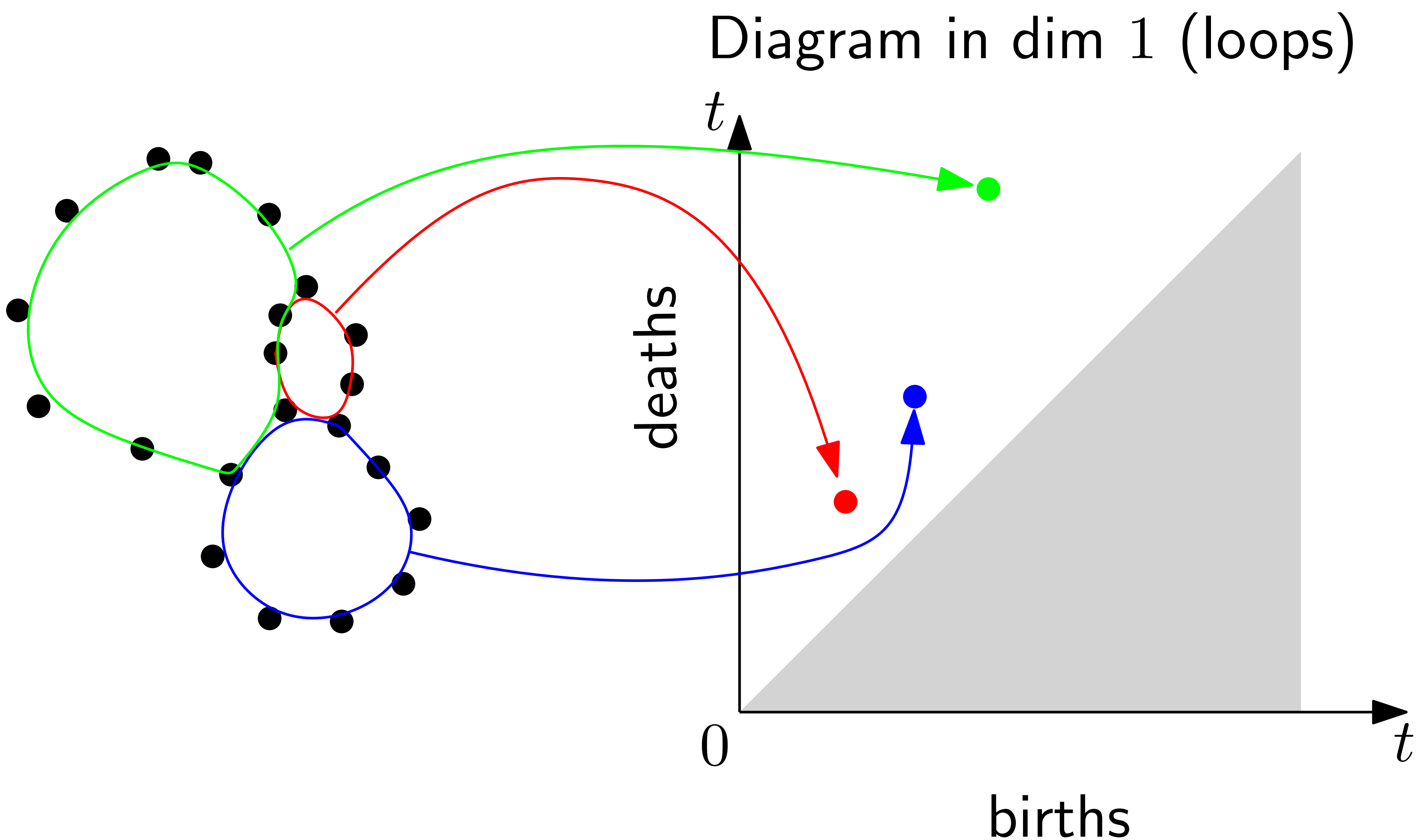
- Compute *at all scale* the homology of the current topological space
- Record *births* and *deaths* of topological features (loop, holes, etc.)
- Print the output as the persistence diagram of the filtration



# The TDA pipeline: persistent homology

- Compute *at all scale* the homology of the current topological space
- Record *births* and *deaths* of topological features (loop, holes, etc.)
- Print the output as the persistence diagram of the filtration

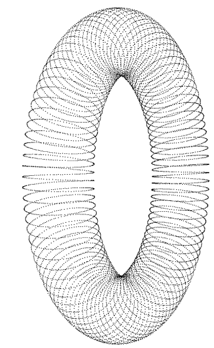


# The TDA pipeline: persistent homology

Summary:

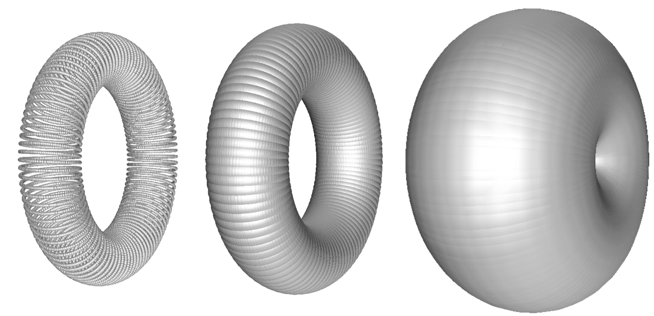
Input item + filtration $f$

$\downarrow$

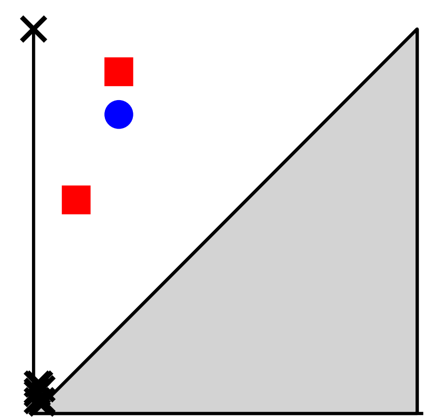


Sequence of increasing topo. spaces

$\downarrow$



Persistence diagram



# The TDA pipeline: persistent homology

Summary:

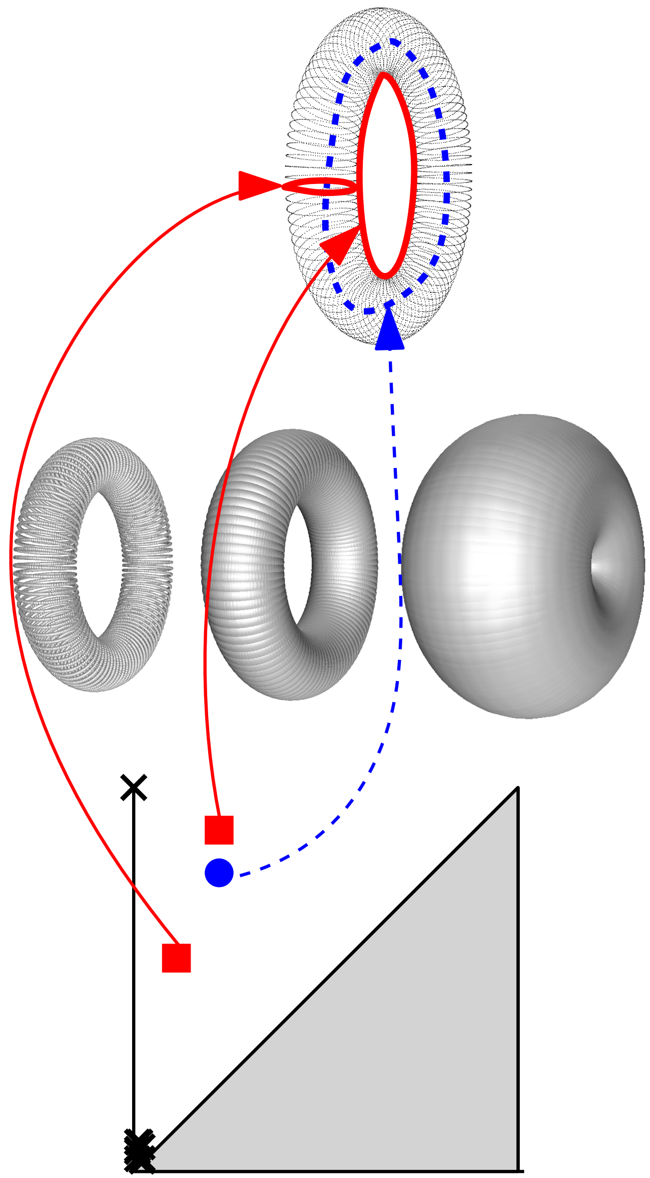
Input item + filtration $f$

↓

Sequence of increasing topo. spaces

↓

Persistence diagram



# Persistence Diagrams as topological features

Given $(X_1 \ldots X_n)$, consider $(\mu_1 \ldots \mu_n)$ their resp. diagrams.

# Persistence Diagrams as topological features

Given $(X_1 \dots X_n)$, consider $(\mu_1 \dots \mu_n)$ their resp. diagrams.

- Can we use the $(\mu_i)_i$s as inputs in a ML task?

$\rightarrow$ PDs are "point clouds" with (possibly) different cardinalities

$\rightarrow$ You can compare PDs with a metric $d_p$ (theoretically motivated)

# Persistence Diagrams as topological features

Given $(X_1 \dots X_n)$, consider $(\mu_1 \dots \mu_n)$ their resp. diagrams.

- Can we use the $(\mu_i)_i$s as inputs in a ML task?

$\rightarrow$ PDs are "point clouds" with (possibly) different cardinalities

$\rightarrow$ You can compare PDs with a metric $d_p$ (theoretically motivated)

**Pbm:** The space of PDs equipped with $d_p$ is ill-structured

no addition

no scalar multiplication

hard to define mean/barycenter

# Persistence Diagrams as topological features

Given $(X_1 \dots X_n)$, consider $(\mu_1 \dots \mu_n)$ their resp. diagrams.

- Can we use the $(\mu_i)_i$s as inputs in a ML task?

$\rightarrow$ PDs are "point clouds" with (possibly) different cardinalities

$\rightarrow$ You can compare PDs with a metric $d_p$ (theoretically motivated)

**Pbm:** The space of PDs equipped with $d_p$ is ill-structured

no addition

no scalar multiplication

hard to define mean/barycenter

**Idea:** Embed PDs into linear spaces via a *feature map* $\Phi$

# Persistence Diagrams as topological features

- Two principal approaches: **finite dim** vectorization
  or infinite dim ones (kernels)

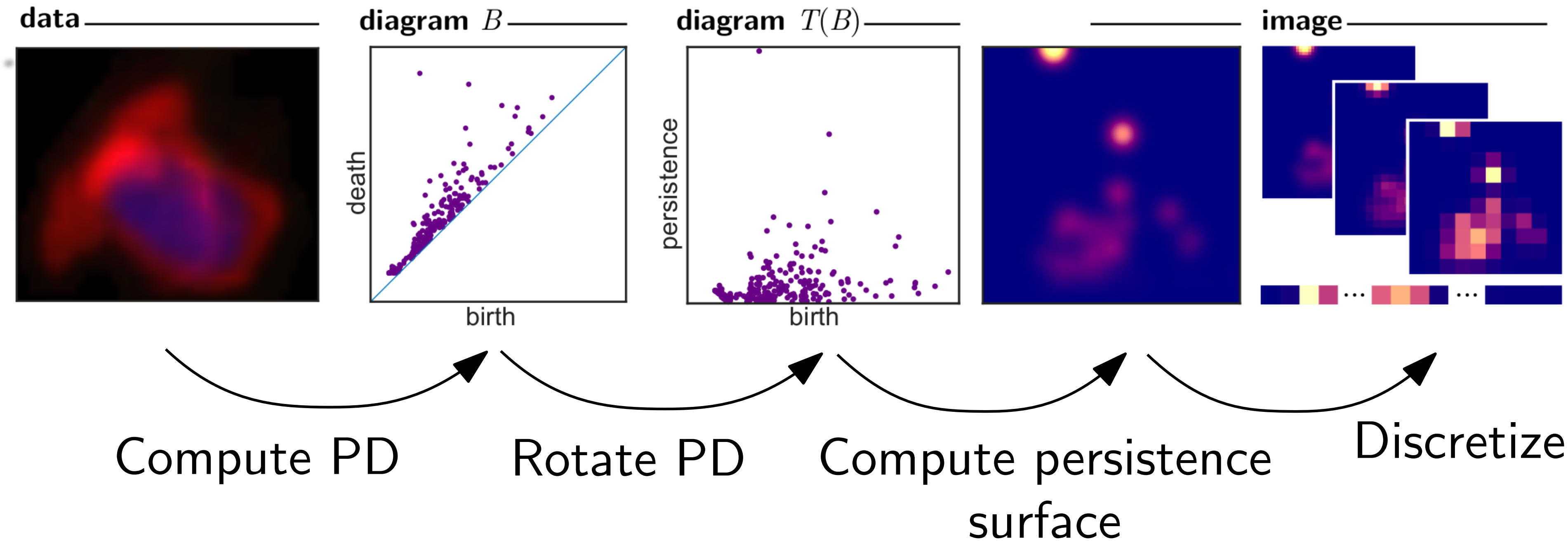
# Persistence Diagrams as topological features

- Two principal approaches: **finite dim** vectorization
  or infinite dim ones (kernels)

Do not scale well on large sets of large diagrams

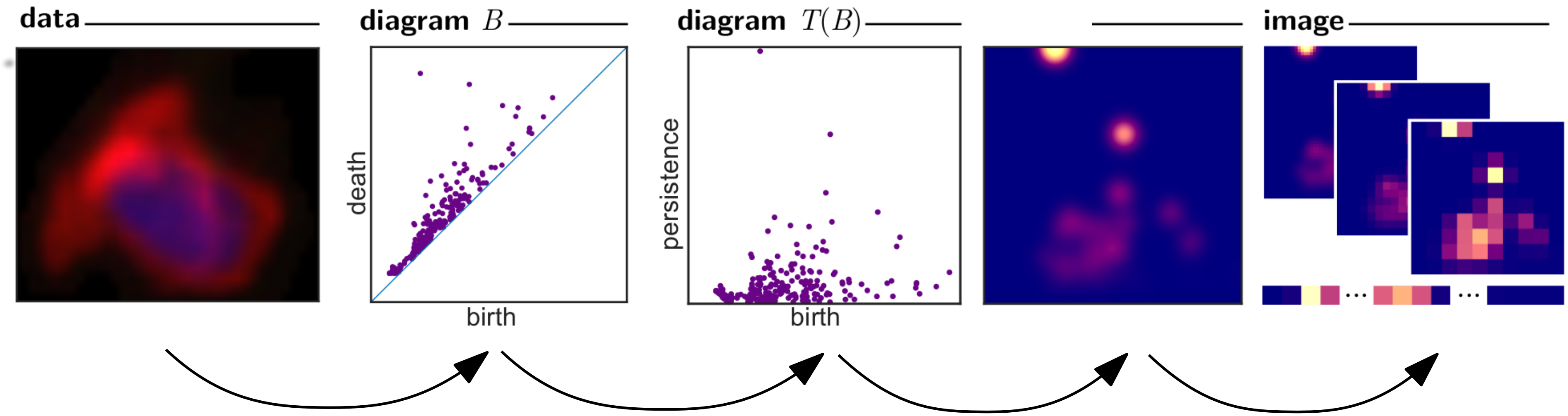
# Persistence Diagrams as topological features

## Persistence Images [Adams et al. 2017]



# Persistence Diagrams as topological features

## Persistence Images [Adams et al. 2017]




Compute PD

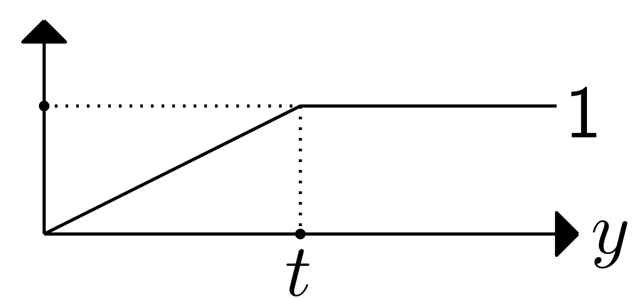
Rotate PD

Compute persistence surface

Discretize

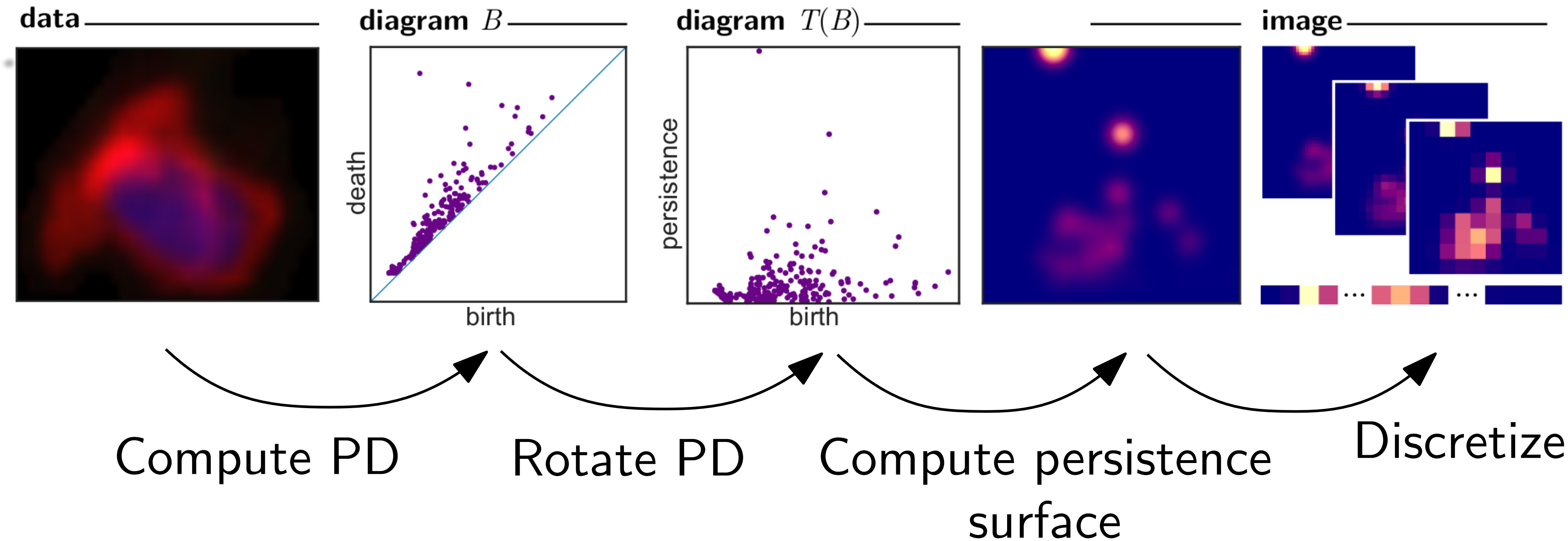
$$\rho_{\color{red}\mu}(\cdot) = \sum_{p \in {\color{red}\mu}} w_t(p) \cdot \exp\left(-\frac{\|\cdot - p\|^2}{2\sigma^2}\right)$$

weight function: $w_t((x, y)) =$

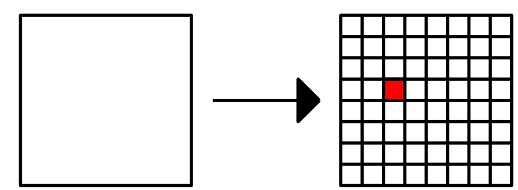


# Persistence Diagrams as topological features

## Persistence Images [Adams et al. 2017]




Discretize plane into grid of pixels:

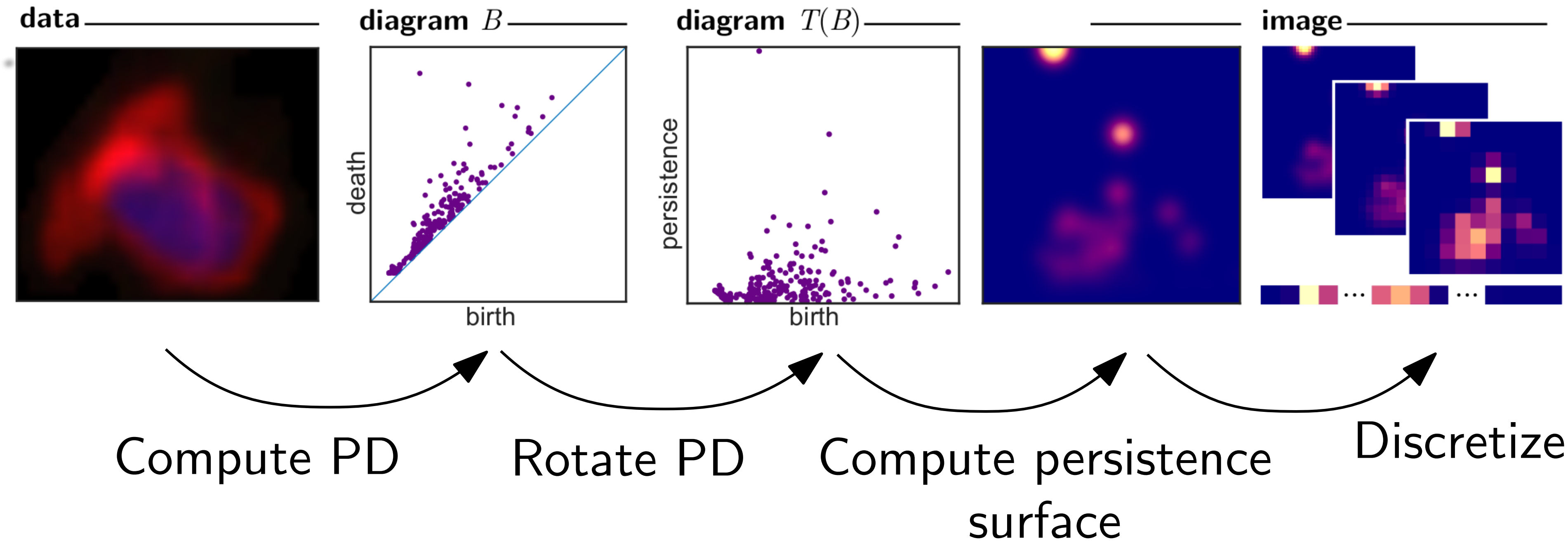


For each pixel $P$, compute $I(P) = \iint_P \rho_\mu$

Concatenate all $I(P)$ into a single vector $\mathrm{PI}(\mu)$ (2D image)

# Persistence Diagrams as topological features

## Persistence Images [Adams et al. 2017]

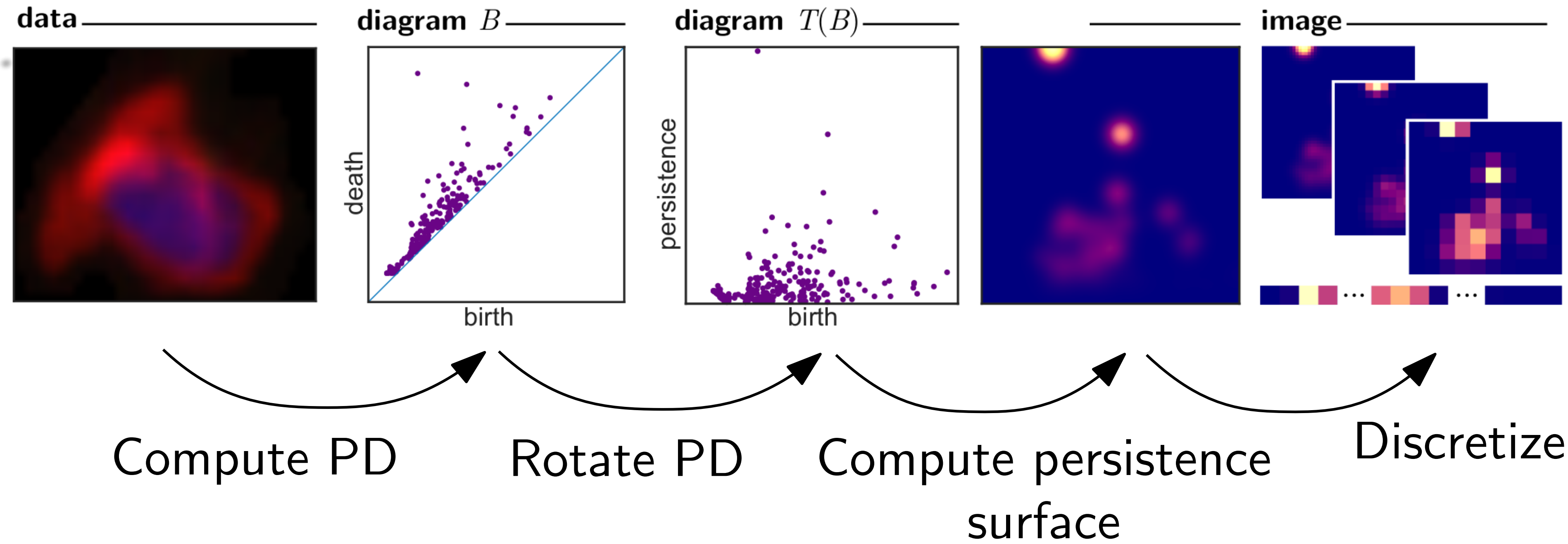



**Prop:** [Adams et al. 2017] (stability)

- $\|\mathrm{PI}(\mu) - \mathrm{PI}(\nu)\|_2 \leq \sqrt{d}C(w, \phi_p)\, d_1(\mu, \nu)$

# Persistence Diagrams as topological features

## Persistence Images [Adams et al. 2017]

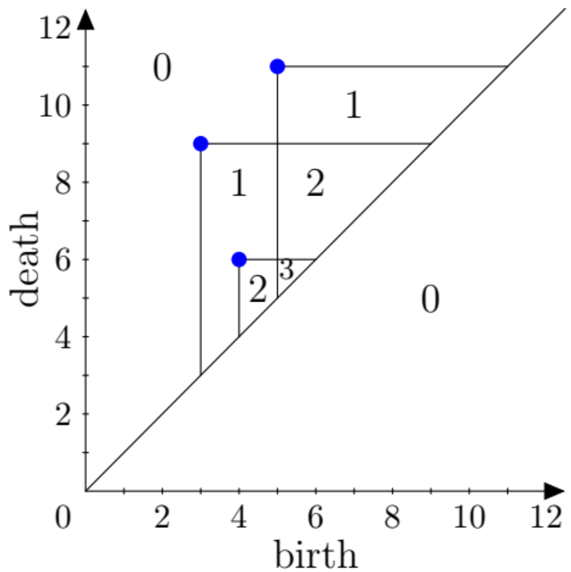


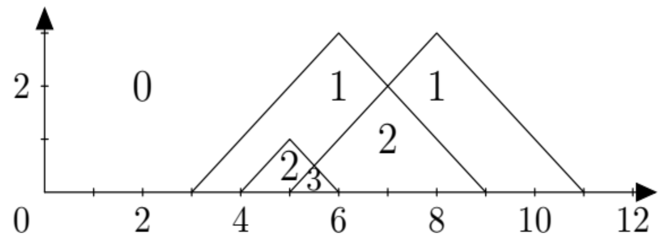

**Observations:** Depends on hyperparams (grid size)

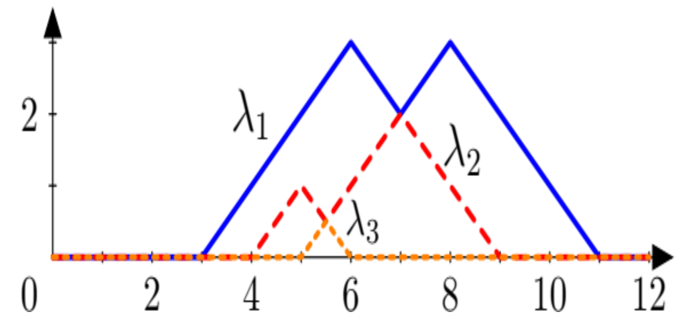
but also trainable params ($\sigma$, $w_t$...)

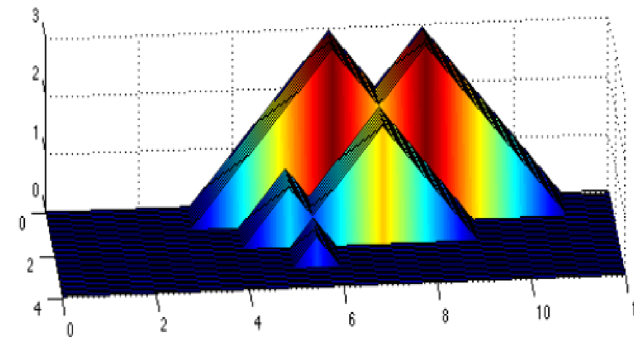
# Persistence Diagrams as topological features

## Persistence Landscape [Bubenik 2015]



Rotate PD

Compute rank function

Use boundaries

of rank function

# Persistence Diagrams as topological features

**Observation:** These vectorizations are of the form

$$\Phi(\mu) = \mathsf{op}\{\phi_\theta(p), p \in \mu\}$$

where op $= \sum, \max, \mathrm{top}-k$, etc.

Where $\theta$ is a set of trainable parameters, and $\phi_\theta : \mathbb{R}^2 \to \mathbb{R}^d$.

**Why?**

# Persistence Diagrams as topological features

**Observation:** These vectorizations are of the form

$$\Phi(\mu) = \mathsf{op}\{\phi_\theta(p), p \in \mu\}$$

where op $= \sum, \max, \mathrm{top}-k$, etc.

Where $\theta$ is a set of trainable parameters, and $\phi_\theta : \mathbb{R}^2 \to \mathbb{R}^d$.

**Why?**

**Idea:** Say a PD is a point cloud $\mu = \{p_1 \dots p_n\} \subset \mathbb{R}^2$.

We want a function $\Phi$ that is *permutation invariant*

$$\Phi(p_1 \dots p_n) = \Phi(p_{\pi(1)} \dots p_\pi(n))$$

for any permutation $\pi \in \mathfrak{S}_n$

# Deep Sets [Zaheer et al. 2017]

**Question:** What are the permutation invariant functions?

# Deep Sets [Zaheer et al. 2017]

**Question:** What are the permutation invariant functions?

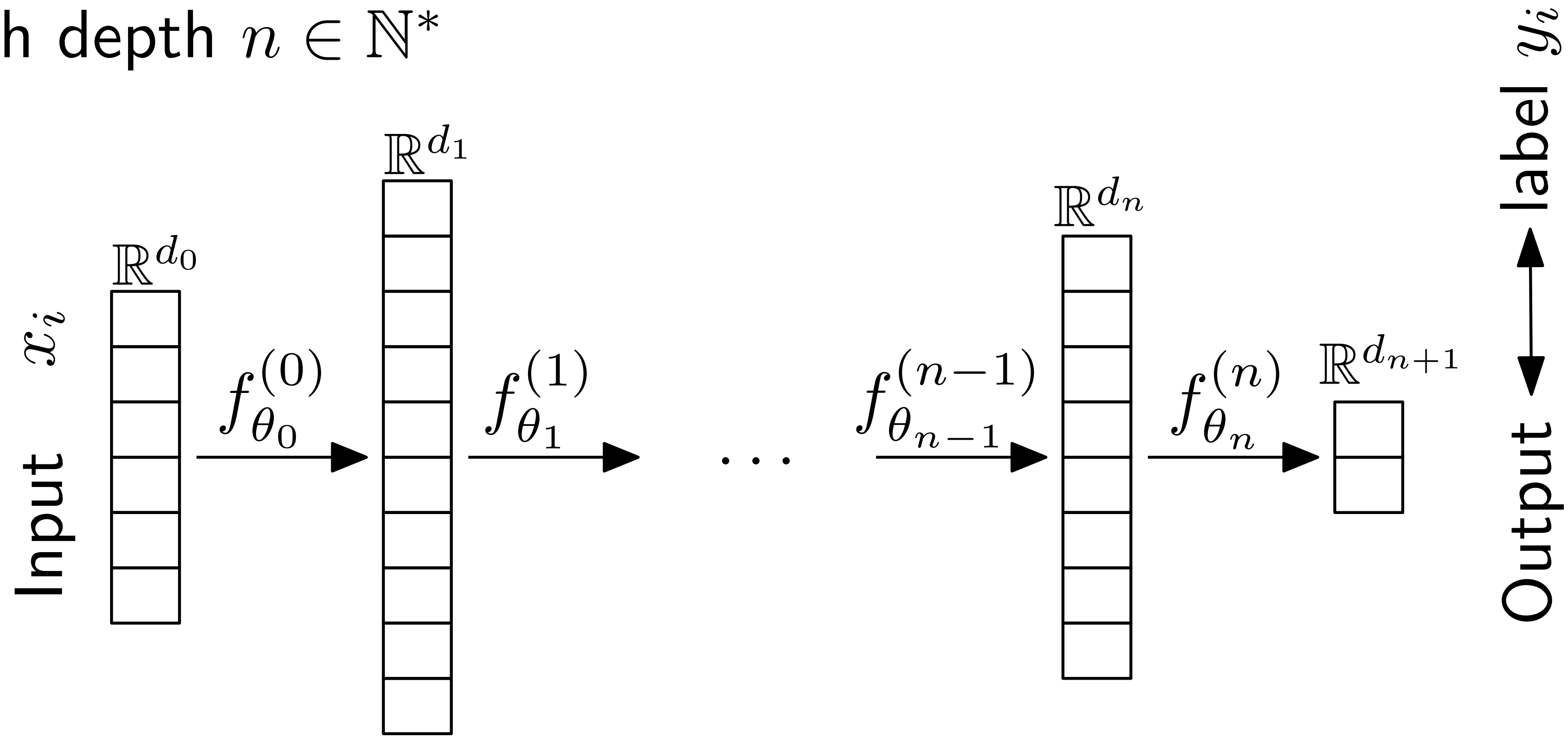
[Zaheer et al. 2017] $\Phi$ is permutation invariant iff $\exists \rho, \phi$ s.t.:

$$\Phi(\{p_1 \dots p_n\}) = \rho\left(\sum_{i=1}^{n} \phi(p_i)\right),$$

provided $(p_i) \subset K$ compact (or countable).

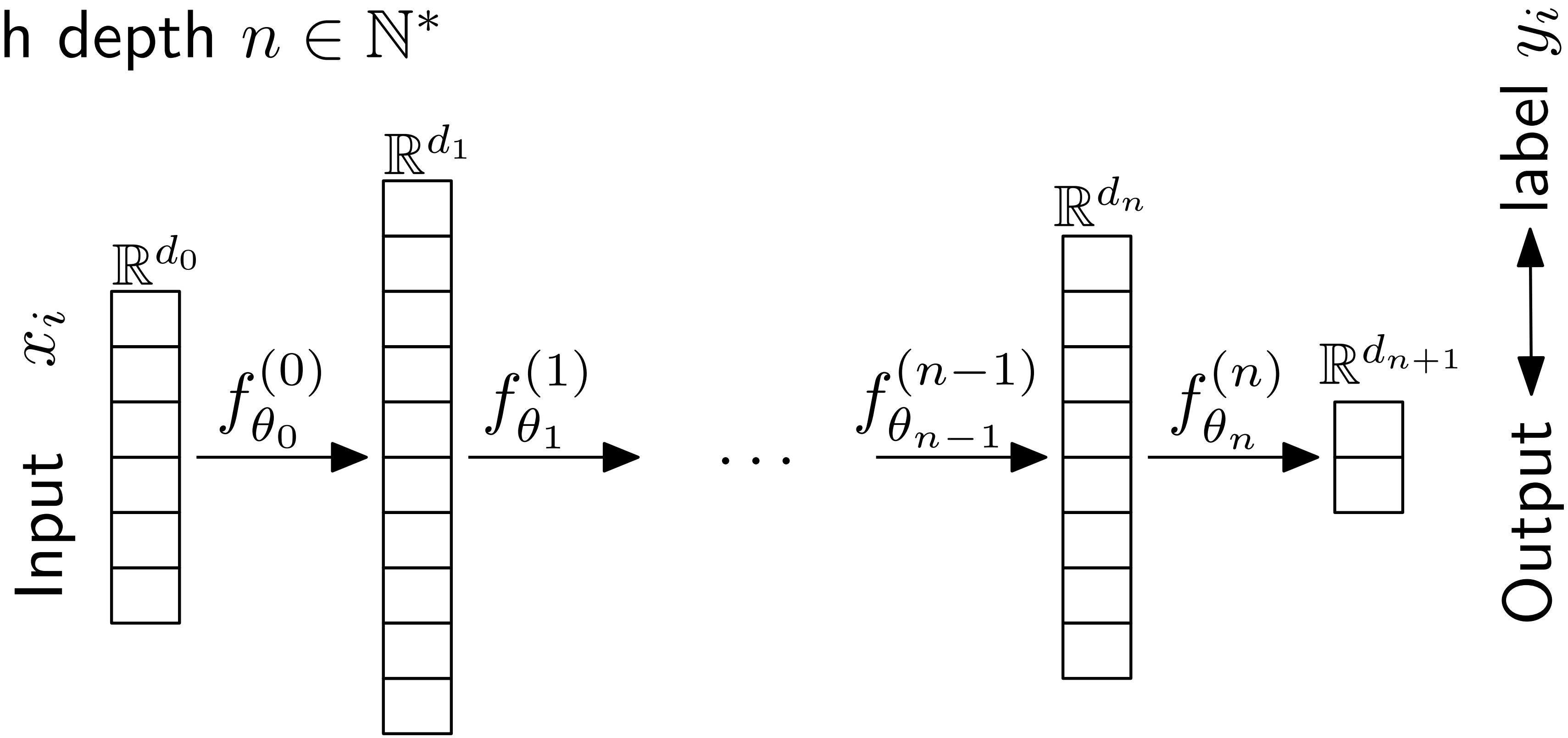
# Reminder: Neural networks

NN with depth $n \in \mathbb{N}^*$



# Reminder: Neural networks

NN with depth $n \in \mathbb{N}^*$

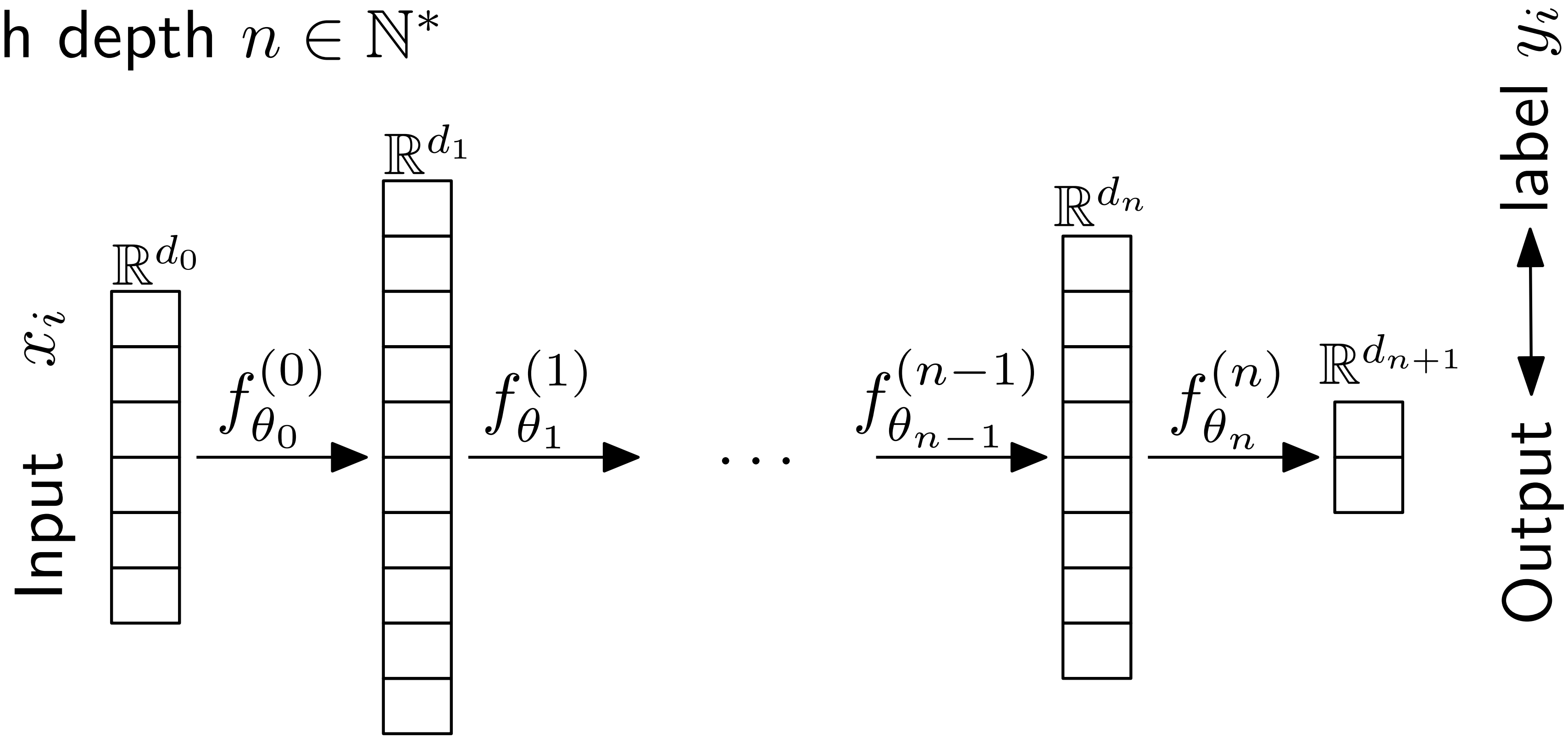



Final predictor: $F_\theta = f^{(n)}_{\theta_n} \circ \cdots \circ f^{(0)}_{\theta_0}$

**Goal:** Minimize $\ell(\theta) = \sum_i \|F_\theta(x_i) - y_i\|_2^2$ w.r.t. $\theta$

# Reminder: Neural networks

NN with depth $n \in \mathbb{N}^*$



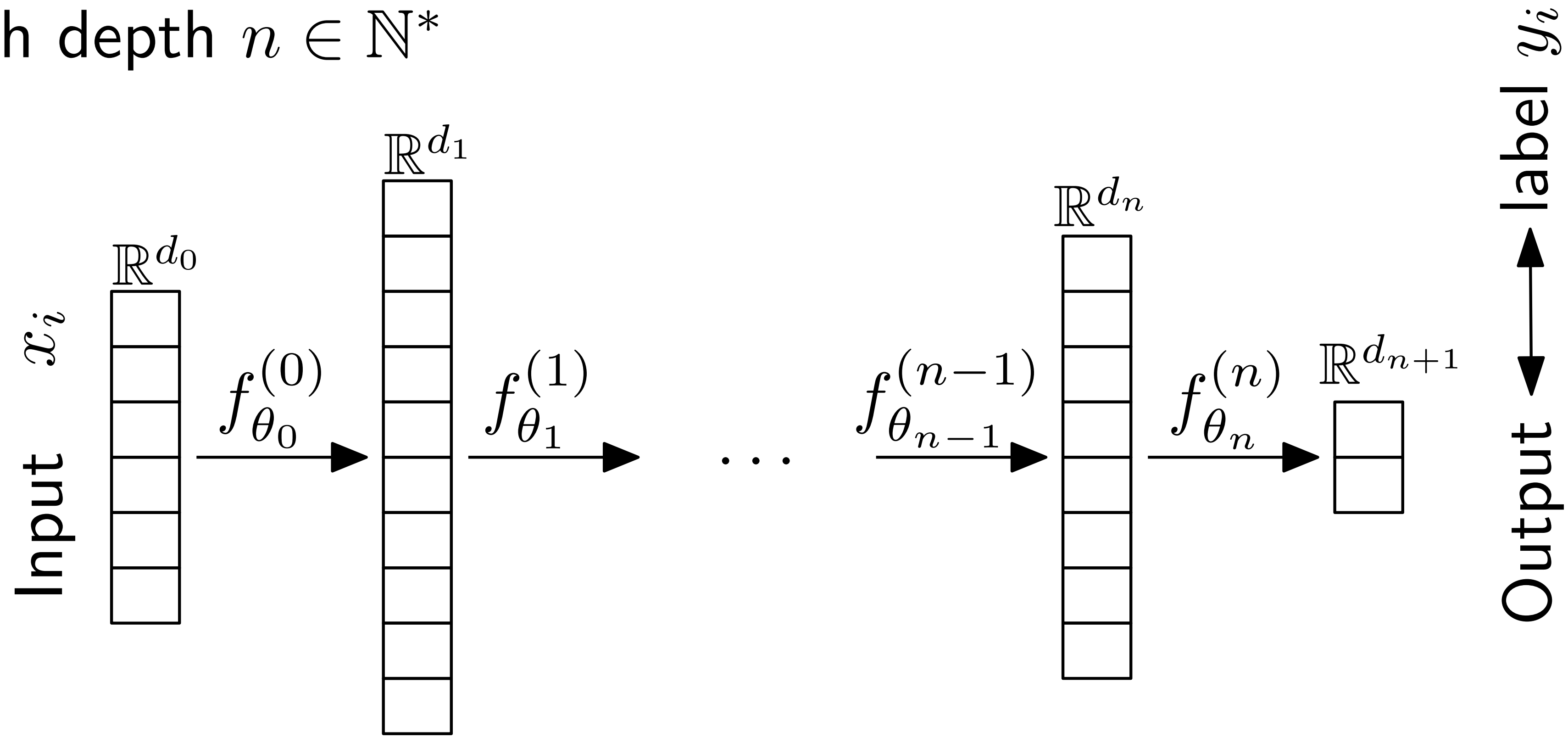

Final predictor: $F_\theta = f^{(n)}_{\theta_n} \circ \cdots \circ f^{(0)}_{\theta_0}$

**Goal:** Minimize $\ell(\theta) = \sum_i \|F_\theta(x_i) - y_i\|_2^2$ w.r.t. $\theta$

**Requirement:** Know $\nabla f^{(i)}_{\theta_i}$ for each $i$. $\Rightarrow$ chain rule.

# Reminder: Neural networks

NN with depth $n \in \mathbb{N}^*$




Final predictor: $F_\theta = f^{(n)}_{\theta_n} \circ \cdots \circ f^{(0)}_{\theta_0}$

**Goal:** Minimize $\ell(\theta) = \sum_i \|F_\theta(x_i) - y_i\|_2^2$ w.r.t. $\theta$

**Requirement:** Know $\nabla f^{(i)}_{\theta_i}$ for each $i$. $\Rightarrow$ chain rule.

**Observation:** If $f^{(0)}_{\theta_0}$ is perm. inv., so is the NN.

## Application to persistence diagrams

Permutation invariant layers generalize several TDA approaches

$\longrightarrow$ persistence lanscapes $\quad$ $\longrightarrow$ silhouettes $\quad$ $\longrightarrow$ Betti curves

## Application to persistence diagrams

Permutation invariant layers generalize several TDA approaches

$\rightarrow$ persistence lanscapes $\rightarrow$ silhouettes $\rightarrow$ Betti curves

$$\mathrm{PersLay}(\mu) = \rho\left(\mathsf{op}\{w(p) \cdot \phi(p)\}_{p\in\mu}\right)$$

Permutation-invariant operation

Weight function

Point transformation

# Application to persistence diagrams

Permutation invariant layers generalize several TDA approaches

$\rightarrow$ persistence lanscapes $\rightarrow$ silhouettes $\rightarrow$ Betti curves

$$\mathrm{PersLay}(\mu) = \rho\left(\mathrm{op}\{w(p)\cdot\phi(p)\}_{p\in\mu}\right)$$

Permutation-invariant operation

Weight function

Point transformation

**Our contribution:** Implementing (and sharing) it.

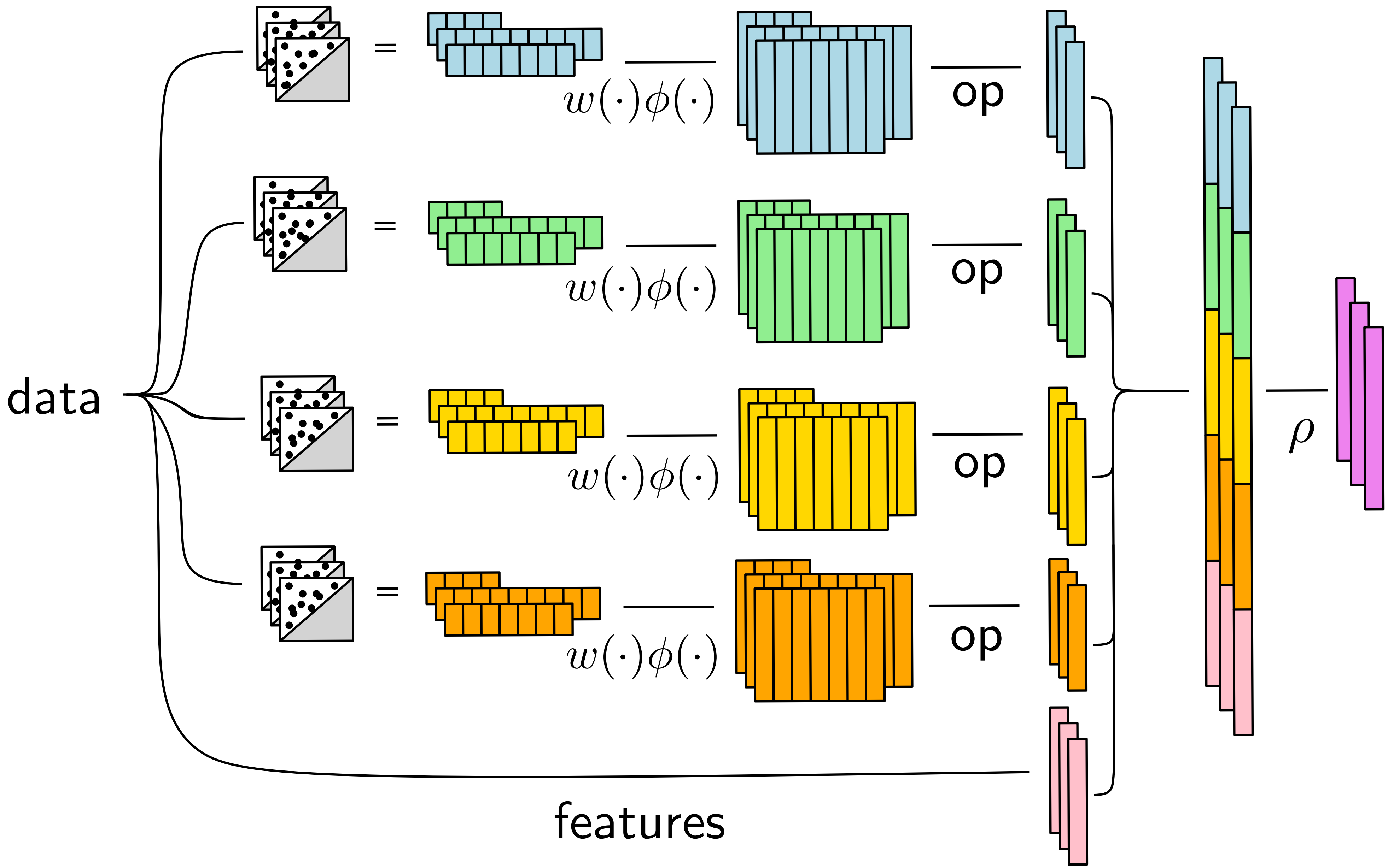
```
from perslay.perslay import perslay_channel
```

```
perslay_parameters["layer"]             = "im"
perslay_parameters["image_size"]        = (20, 20)
```

```
perslay_parameters["perm_op"] = "sum"
```

```
perslay_channel(output  =  list_v,                    # outputs
                name    =  "perslay",                 # name of this layer
                diag    =  YOUR_DIAGS,                # diagrams
                **self.perslay_parameters)
```

# Application to persistence diagrams

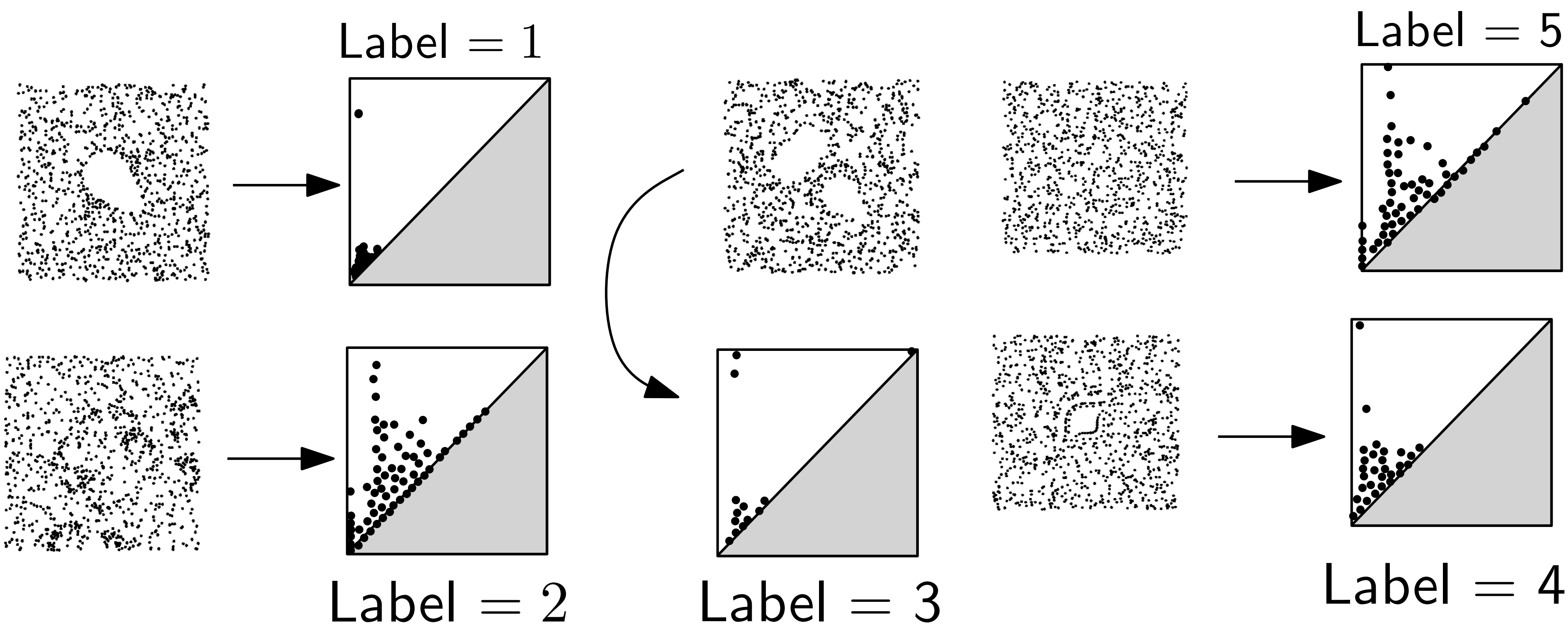


# Application to persistence diagrams

**Goal**: classify orbits of *linked twisted map* (flow dynamics)

Orbits described by (depending on parameter $r$):

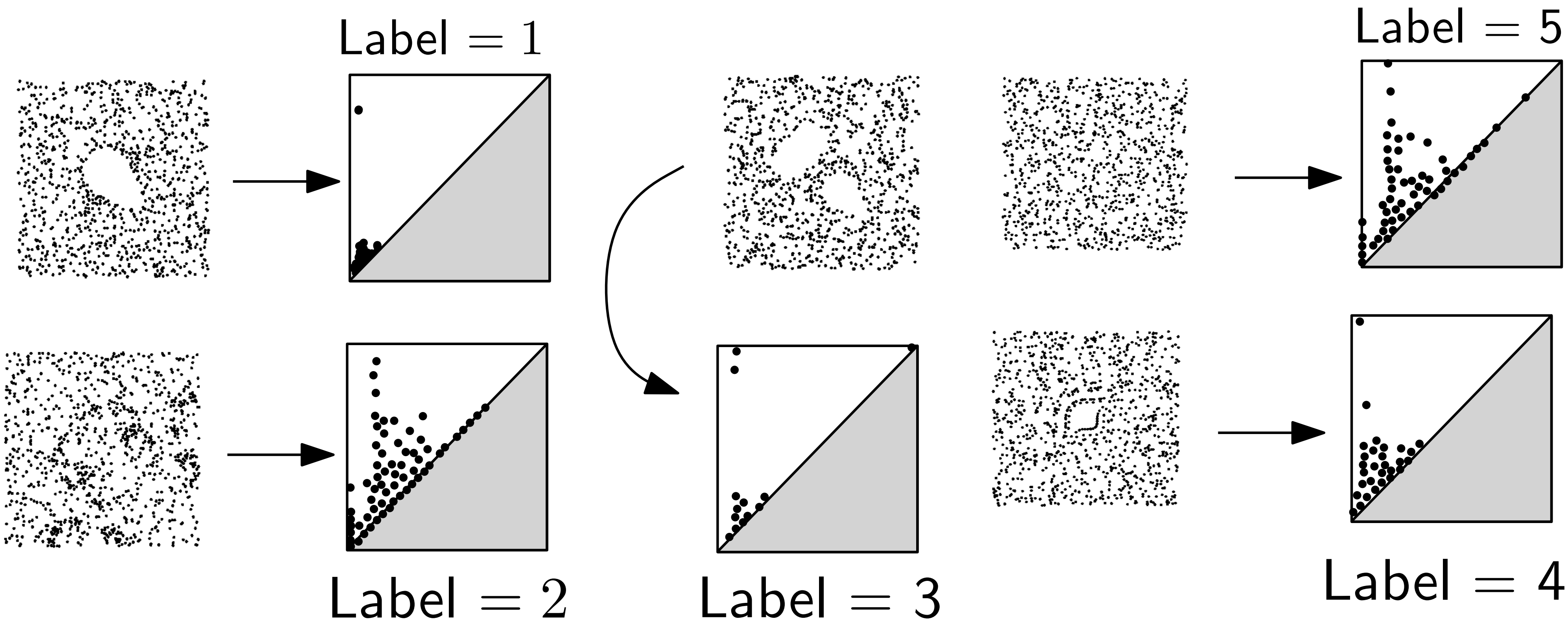
$$\begin{cases} x_{n+1} = x_n + r\, y_n(1 - y_n) \mod 1 \\ y_{n+1} = y_n + r\, x_{n+1}(1 - x_{n+1}) \mod 1 \end{cases}$$



# Application to persistence diagrams

**Goal**: classify orbits of *linked twisted map* (flow dynamics)

| Dataset | PSS-K | PWG-K | SW-K | PF-K | PersLay |
|---|---|---|---|---|---|
| ORBIT5K | 72.38(±2.4) | 76.63(±0.7) | 83.6(±0.9) | 85.9(±0.8) | **87.7(±1.0)** |
| ORBIT100K | — | — | — | — | **89.2(±0.3)** |



## Take home messages

- Topological Data Analysis builds topological summary of a given object: the persistence diagram.
- PDs are not straightforward to use in ML pipelines
- A workaround is to vectorize our PDs, adaptatively to a given learning task (e.g. classification).
- The "Deep Sets" architecture can be used to incorporate PDs in NN architecture while encompassing existing vectorizations.

## Take home messages

- Topological Data Analysis builds topological summary
  of a given object: the persistence diagram.
- PDs are not straightforward to use in ML pipelines
- A workaround is to vectorize our PDs, adaptatively
  to a given learning task (e.g. classification).
- The "Deep Sets" architecture can be used to incorporate PDs in NN architecture while encompassing existing vectorizations.

## Supplementary material

- *PersLay: A Neural Network Layer for Persistence Diagrams and New Graph Topological Signatures*, AISTATS 2020
- `https://github.com/MathieuCarriere/perslay`
  $\rightarrow$ soon integrated to the `Gudhi` library (hopefully).