Multiparameter Persistence and Nonlinear Hierarchical Clustering

Oliver Gäfvert

KTH Royal Institute of Technology/ICERM

oliverg@kth.se

Multiparameter Persistence and Nonlinear Hierarchical Clustering

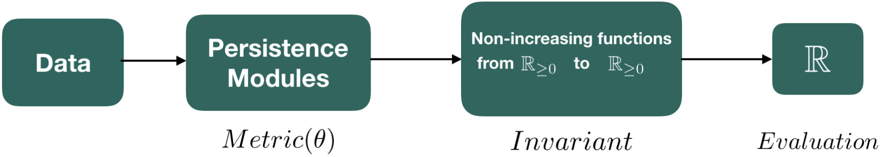
Oliver Gäfvert

KTH Royal Institute of Technology/ICERM

oliverg@kth.se

Contour Learning

Contour Learning



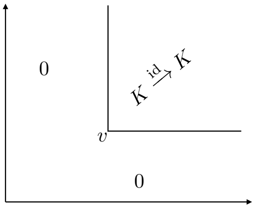
Preliminaries

Persistence modules are functors $F\colon \mathbb{R}^n \to \mathrm{Vect}_K$.

Preliminaries

Persistence modules are functors $F\colon \mathbb{R}^n \to \mathrm{Vect}_K$.

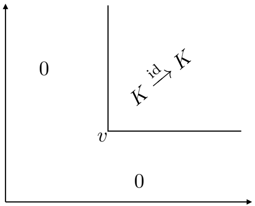
The free functor $K(v,-)\colon \mathbb{R}^2 \to \mathrm{Vect}_K$ on one generator:



Preliminaries

Persistence modules are functors $F\colon \mathbb{R}^n \to \mathrm{Vect}_K$.

The free functor $K(v,-)\colon \mathbb{R}^2 \to \mathrm{Vect}_K$ on one generator:

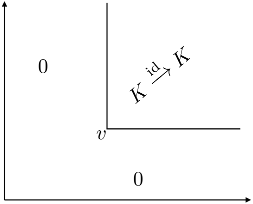


- F is *finitely gen.* if there is a surjection $\oplus_{i=1}^m K(v_i,-) \overset{\phi}{\twoheadrightarrow} F$.

Preliminaries

Persistence modules are functors $F\colon \mathbb{R}^n \to \mathsf{Vect}_K$.

The free functor $K(v,-)\colon \mathbb{R}^2 \to \mathsf{Vect}_K$ on one generator:

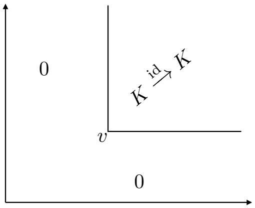


- F is *finitely gen.* if there is a surjection $\oplus_{i=1}^m K(v_i,-) \stackrel{\phi}{\twoheadrightarrow} F$.
- The minimal such m is called the *rank* of F.

Preliminaries

Persistence modules are functors $F\colon \mathbb{R}^n \to \mathsf{Vect}_K$.

The free functor $K(v,-)\colon \mathbb{R}^2 \to \mathsf{Vect}_K$ on one generator:



- F is *finitely gen.* if there is a surjection $\oplus_{i=1}^m K(v_i,-) \overset{\phi}{\twoheadrightarrow} F$.
- The minimal such m is called the *rank* of F.
- F is *finitely presented* if it is f.g and the kernel of ϕ is also f.g.

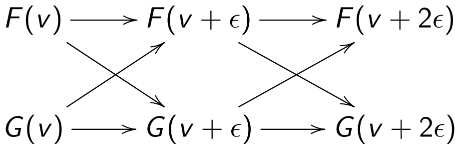
Metrics on Persistence Modules

The most common distance is the **interleaving distance**.

Metrics on Persistence Modules

The most common distance is the **interleaving distance**.

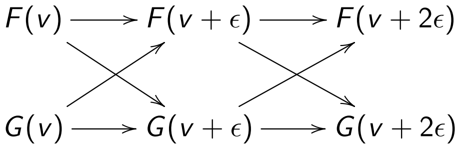
F and G are *ϵ-interleaved* if there are maps s.t the following diagram commutes for all v:



Metrics on Persistence Modules

The most common distance is the **interleaving distance**.

F and G are *ϵ-interleaved* if there are maps s.t the following diagram commutes for all v:



$$d_{\bowtie} = \inf\{\epsilon \mid F \text{ and } G \text{ are } \epsilon\text{-interleaved}\}$$

Multiparameter Persistent Homology

The categories of:

- *finitely presented persistence modules*
- *f.g n-graded $K[x_1, \ldots, x_n]$-modules*

have the same formal properties:

Multiparameter Persistent Homology

The categories of:

- *finitely presented persistence modules*
- *f.g n-graded* $K[x_1, \ldots, x_n]$*-modules*

have the same formal properties:

- *enough projectives*

Multiparameter Persistent Homology

The categories of:

- *finitely presented persistence modules*
- *f.g n-graded* $K[x_1, \ldots, x_n]$*-modules*

have the same formal properties:

- *enough projectives*
- *all projectives are free*

Multiparameter Persistent Homology

The categories of:

- *finitely presented persistence modules*
- *f.g n-graded* $K[x_1, \ldots, x_n]$*-modules*

have the same formal properties:

- *enough projectives*
- *all projectives are free*
- *any object has a minimal resolution of length at most n.*

Multiparameter Persistent Homology

The categories of:

- *finitely presented persistence modules*
- *f.g n-graded $K[x_1, \ldots, x_n]$-modules*

have the same formal properties:

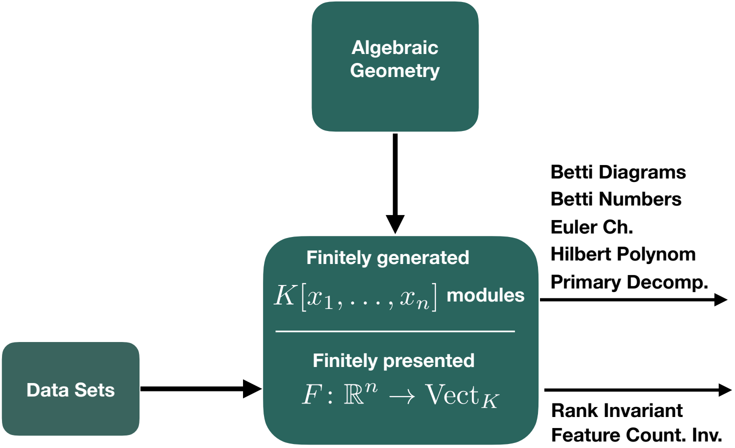
- *enough projectives*
- *all projectives are free*
- *any object has a minimal resolution of length at most n.*

Slogan

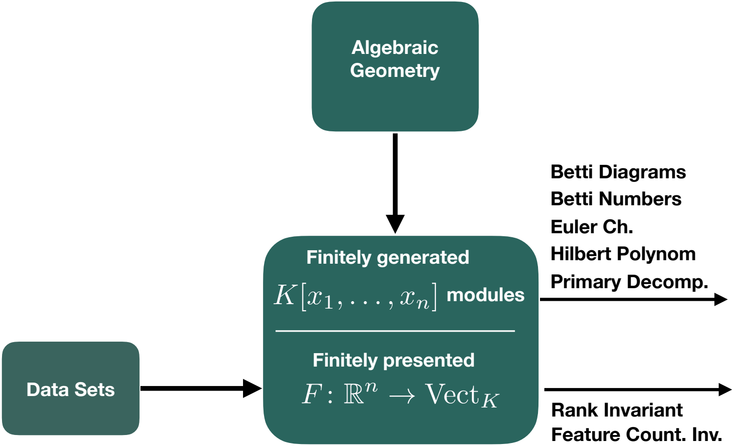
We can identify a finitely presented persistence module with an n-graded $K[x_1, \ldots, x_n]$-module by restricting to a small enough grid $\mathbb{N}^n \subset \mathbb{R}^n$.

Invariants

Invariants


Algebraic Geometry
Betti Diagrams
Betti Numbers
Euler Ch.
Hilbert Polynom
Primary Decomp.
Finitely generated
$K[x_1, \ldots, x_n]$ modules
Finitely presented
$F\colon \mathbb{R}^n \to \mathrm{Vect}_K$
Data Sets
Rank Invariant
Feature Count. Inv.

Invariants



Need for stable invariants!

Feature Counting Invariant

- Introduced by Scolamiero et. al. in [4].

Feature Counting Invariant

- Introduced by Scolamiero et. al. in [4].
- $\widehat{\beta_0}(F)(\tau) = \min\{\mathrm{rank}(G) \mid d(F, G) \leq \tau\}$

Feature Counting Invariant

- Introduced by Scolamiero et. al. in [4].
- $\widehat{\beta}_0(F)(\tau) = \min\{\mathrm{rank}(G) \mid d(F, G) \leq \tau\}$
- Can be seen as a stabilization of a classical invariant, namely the zeroth Betti number.

Feature Counting Invariant

- Introduced by Scolamiero et. al. in [4].
- $\widehat{\beta}_0(F)(\tau) = \min\{\text{rank}(G) \mid d(F, G) \leq \tau\}$
- Can be seen as a stabilization of a classical invariant, namely the zeroth Betti number.
- Depends strongly on the underlying metric.

Feature Counting Invariant

- Introduced by Scolamiero et. al. in [4].
- $\widehat{\beta_0}(F)(\tau) = \min\{\mathrm{rank}(G) \mid d(F, G) \leq \tau\}$
- Can be seen as a stabilization of a classical invariant, namely the zeroth Betti number.
- Depends strongly on the underlying metric. (This is what we will use.)

Feature Counting Invariant

- Introduced by Scolamiero et. al. in [4].
- $\widehat{\beta}_0(F)(\tau) = \min\{\text{rank}(G) \mid d(F, G) \leq \tau\}$
- Can be seen as a stabilization of a classical invariant, namely the zeroth Betti number.
- Depends strongly on the underlying metric. (This is what we will use.)

Using *persistence contours*, we can show:

Theorem (G. and Chachólski, 2017, [2])

For $n \geq 2$, computing $\widehat{\beta}_0(F)$ is NP-hard.

Persistence Contours

(Metrics on persistence modules)

Persistence Contours

A *persistence contour* is a functor $C\colon \mathbb{R}^n_\infty \times \mathbb{R} \to \mathbb{R}^n_\infty$ s.t, for any $v \in \mathbb{R}^n_\infty$ and any $\epsilon, \tau \in \mathbb{R}$:

1. $v \leq C(v, \epsilon)$,
2. $C(C(v, \epsilon), \tau) \leq C(v, \epsilon + \tau)$.

Persistence Contours

A *persistence contour* is a functor $C\colon \mathbb{R}^n_\infty \times \mathbb{R} \to \mathbb{R}^n_\infty$ s.t, for any $v \in \mathbb{R}^n_\infty$ and any $\epsilon, \tau \in \mathbb{R}$:

1. $v \leq C(v, \epsilon)$,
2. $C(C(v, \epsilon), \tau) \leq C(v, \epsilon + \tau)$.

Similar to superlinear families by Bubenick et. al. [1].

Persistence Contours

A *persistence contour* is a functor $C\colon \mathbb{R}^n_\infty \times \mathbb{R} \to \mathbb{R}^n_\infty$ s.t, for any $v \in \mathbb{R}^n_\infty$ and any $\epsilon, \tau \in \mathbb{R}$:

1. $v \leq C(v, \epsilon)$,
2. $C(C(v, \epsilon), \tau) \leq C(v, \epsilon + \tau)$.

Similar to superlinear families by Bubenick et. al. [1].

- The *ϵ-shift*,

  $$F[\epsilon] = \langle \{F(v_i \leq C(v_i, \epsilon))(g_i) \in F(C(v_i, \epsilon))\} \rangle$$

  for any generating set $\{g_i \in F(v_i)\}$ of F.

A *persistence contour* is a functor $C\colon \mathbb{R}^n_\infty \times \mathbb{R} \to \mathbb{R}^n_\infty$ s.t, for any $v \in \mathbb{R}^n_\infty$ and any $\epsilon, \tau \in \mathbb{R}$:

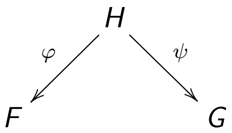
1. $v \leq C(v, \epsilon)$,
2. $C(C(v, \epsilon), \tau) \leq C(v, \epsilon + \tau)$.

Similar to superlinear families by Bubenick et. al. [1].

- The *ϵ-shift*,

$$F[\epsilon] = \langle \{F(v_i \leq C(v_i, \epsilon))(g_i) \in F(C(v_i, \epsilon))\} \rangle$$

  for any generating set $\{g_i \in F(v_i)\}$ of F.

- 

  F and G are *ϵ-close* if $\ker \varphi[a] = \operatorname{coker} \varphi[b] = \ker \psi[c] =$
  $= \operatorname{coker} \psi[d] = 0$ and $a + b + c + d \leq \epsilon$.

Persistence Contours

Given a choice of persistence contour:

$$d(F, G) := \inf\{\epsilon \mid F \text{ and } G \text{ are } \epsilon\text{-close}\}$$

Persistence Contours

Given a choice of persistence contour:

$$d(F, G) := \inf\{\epsilon \mid F \text{ and } G \text{ are } \epsilon\text{-close}\}$$

$\implies$ a class of extended pseudometrics on persistence modules.

Persistence Contours

Given a choice of persistence contour:

$$d(F, G) := \inf\{\epsilon \mid F \text{ and } G \text{ are } \epsilon\text{-close}\}$$

$\Longrightarrow$ a class of extended pseudometrics on persistence modules.

Example

Let $C(v, \epsilon) = v + \epsilon\mathbf{1}$. This is called the *standard persistence contour*.

Persistence Contours

Given a choice of persistence contour:

$$d(F, G) := \inf\{\epsilon \mid F \text{ and } G \text{ are } \epsilon\text{-close}\}$$

$\Longrightarrow$ a class of extended pseudometrics on persistence modules.

Example

Let $C(v, \epsilon) = v + \epsilon\mathbf{1}$. This is called the *standard persistence contour*.

Theorem

For the standard persistence contour:

$$\frac{1}{6} d(F, G) \leq d_{\bowtie}(F, G) \leq d(F, G)$$

Constructing Persistence Contours

Constructing Persistence Contours

We define $C_\theta \colon \mathbb{R}_\infty \times \mathbb{R} \to \mathbb{R}_\infty$ as the solution to the following equation:

$$t = \int_{v}^{C_\theta(v,t)} \rho_\theta(\alpha) d\alpha$$

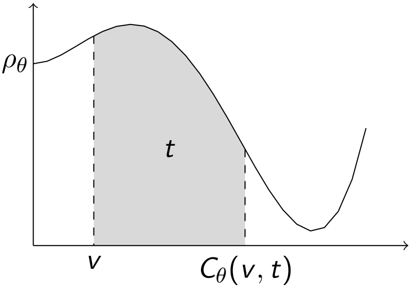
where $\rho_\theta(\alpha)$ is some positive real-valued function, parametrized by $\theta \in \mathbb{R}^m$.

Constructing Persistence Contours

We define $C_\theta \colon \mathbb{R}_\infty \times \mathbb{R} \to \mathbb{R}_\infty$ as the solution to the following equation:

$$t = \int_v^{C_\theta(v,t)} \rho_\theta(\alpha) d\alpha$$

where $\rho_\theta(\alpha)$ is some positive real-valued function, parametrized by $\theta \in \mathbb{R}^m$.



Constructing Persistence Contours

The kernel function ρ_θ can be chosen as:

- step-function
- neural network
- mixture of gaussians

Constructing Persistence Contours

The kernel function ρ_θ can be chosen as:

- step-function
- neural network
- mixture of gaussians

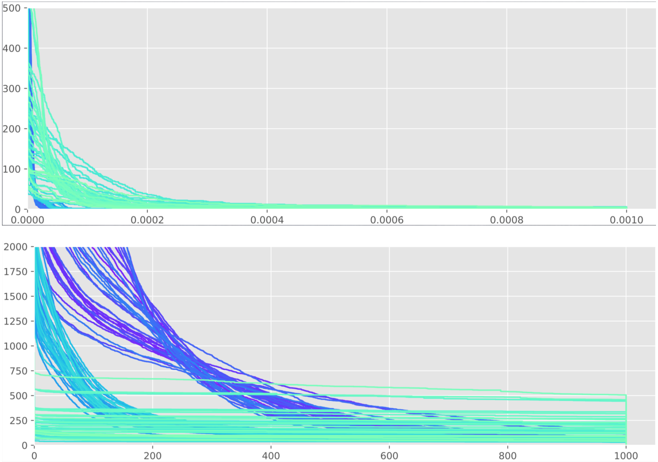
We consider persistence contours parametrized by $\theta \in \mathbb{R}^m$, with the property:

$$C_\theta(v, t) = (C_\theta^1(v_1, t), \ldots, C_\theta^n(v_n, t))$$

where $v = (v_1, \ldots, v_n)$.

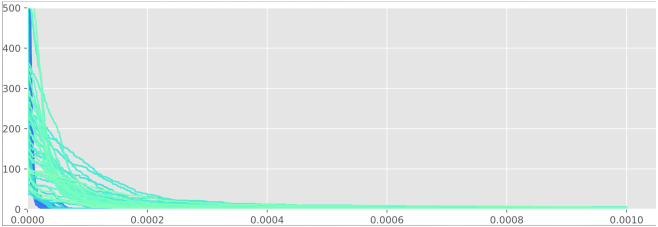
Tree Contour Example

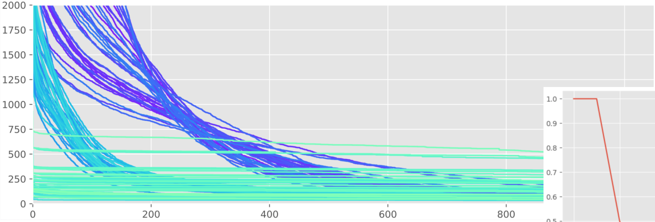
Forest tree data from Henri Riihimäki:

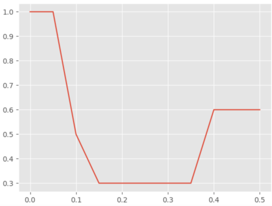


The feature counting function for two different kinds of contours on the forest tree dataset.

Forest tree data from Henri Riihimäki:







The feature counting function for two different kinds of contours on the fores

Contour Learning

Contour Learning

Problem

Contour Learning

Problem

Given the following information:

Contour Learning

Problem

Given the following information:

1. a set of persistence modules $\mathcal{F} = \{F_1, \ldots, F_n\}$,

Contour Learning

Problem

Given the following information:

1. a set of persistence modules $\mathcal{F} = \{F_1, \ldots, F_n\}$,
2. a map $\phi \colon \mathcal{F} \to \{0, 1\}$, assigning a class to each module,

Contour Learning

Problem

Given the following information:

1. a set of persistence modules $\mathcal{F} = \{F_1, \ldots, F_n\}$,
2. a map $\phi \colon \mathcal{F} \to \{0, 1\}$, assigning a class to each module,

we want to find a (pseudo)metric that discriminates the modules of different classes.

Contour Learning

Problem

Given the following information:

1. a set of persistence modules $\mathcal{F} = \{F_1, \dots, F_n\}$,
2. a map $\phi\colon \mathcal{F} \to \{0, 1\}$, assigning a class to each module,

we want to find a (pseudo)metric that discriminates the modules of different classes.

Solution

Contour Learning

Problem

Given the following information:

1. a set of persistence modules $\mathcal{F} = \{F_1, \dots, F_n\}$,
2. a map $\phi \colon \mathcal{F} \to \{0, 1\}$, assigning a class to each module,

we want to find a (pseudo)metric that discriminates the modules of different classes.

Solution

Define a distance d_θ, using persistence contours, and solve the following metric learning problem:

$$\begin{aligned} \min_\theta \sum_{F, F' \in \phi^{-1}(0)} d_\theta(F, F') + \sum_{F, F' \in \phi^{-1}(1)} d_\theta(F, F') \\ \text{s.t} \sum_{\substack{F \in \phi^{-1}(0) \\ F' \in \phi^{-1}(1)}} d_\theta(F, F') \geq 1 \end{aligned} \tag{1}$$

One-parameter contour learning

When $n = 1$, F has a unique bar decomposition B.

One-parameter contour learning

When $n = 1$, F has a unique bar decomposition B.

Theorem

For $n = 1$, the feature counting function is computed as:
$\widehat{\beta}_0(F)(\tau) = \#\{[a, b] \in B \mid b > C(a, \tau)\}$

One-parameter contour learning

When $n = 1$, F has a unique bar decomposition B.

Theorem

For $n = 1$, the feature counting function is computed as:
$\widehat{\beta_0}(F)(\tau) = \#\{[a, b] \in B \mid b > C(a, \tau)\}$

Sketch of proof.

When $n = 1$, the bar decomposition of $F[\tau]$ consists of
$B_\tau = \{[C(a, \tau), b] \mid [a, b] \in B \text{ and } C(a, \tau) < b\}$.

One-parameter contour learning

When $n = 1$, F has a unique bar decomposition B.

Theorem

For $n = 1$, the feature counting function is computed as:
$\widehat{\beta_0}(F)(\tau) = \#\{[a, b] \in B \mid b > C(a, \tau)\}$

Sketch of proof.

When $n = 1$, the bar decomposition of $F[\tau]$ consists of
$B_\tau = \{[C(a, \tau), b] \mid [a, b] \in B \text{ and } C(a, \tau) < b\}$.

- Claim: any τ-close functor to F contains $F[\tau]$.

One-parameter contour learning

When $n = 1$, F has a unique bar decomposition B.

Theorem

For $n = 1$, the feature counting function is computed as:
$\widehat{\beta_0}(F)(\tau) = \#\{[a, b] \in B \mid b > C(a, \tau)\}$

Sketch of proof.

When $n = 1$, the bar decomposition of $F[\tau]$ consists of $B_\tau = \{[C(a, \tau), b] \mid [a, b] \in B \text{ and } C(a, \tau) < b\}$.

- Claim: any τ-close functor to F contains $F[\tau]$.
- If $H \supset F[\tau]$, then $\text{rank}(H) \geq \text{rank}(F[\tau])$.

□

One-parameter contour learning

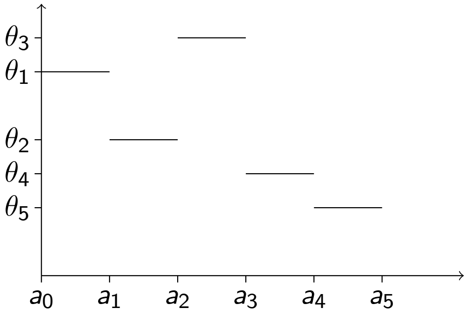
Define the kernel function

Choose ρ_θ as a step-function, where $\rho_\theta(\alpha) = \theta_i$ if $a_{i-1} \leq \alpha < a_i$.

One-parameter contour learning

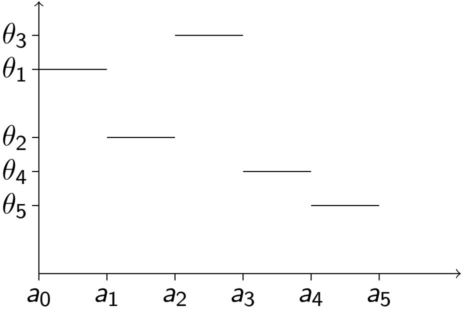
Define the kernel function

Choose ρ_θ as a step-function, where $\rho_\theta(\alpha) = \theta_i$ if $a_{i-1} \leq \alpha < a_i$.



One-parameter contour learning

Define the kernel function
Choose ρ_θ as a step-function, where $\rho_\theta(\alpha) = \theta_i$ if $a_{i-1} \leq \alpha < a_i$.



we will adjust the step heights θ_i to solve the metric learning problem (1).

One-parameter contour learning

Define a smoothed version of the feature counting invariant

The feature counting invariant is a step function, and we want to make it smooth.

One-parameter contour learning

Define a smoothed version of the feature counting invariant

The feature counting invariant is a step function, and we want to make it smooth.Define

$$\mathfrak{r}_\theta(F)(t) := \sum_{[a,b]\in B} \frac{1}{1 + e^{k(C_\theta(a,t)-b)}}$$

where B is the bar decomposition of F.

One-parameter contour learning

Define a smoothed version of the feature counting invariant

The feature counting invariant is a step function, and we want to make it smooth.Define

$$\mathfrak{r}_\theta(F)(t) := \sum_{[a,b]\in B} \frac{1}{1 + e^{k(C_\theta(a,t)-b)}}$$

where B is the bar decomposition of F.

Choose the metric on persistence modules

Smoothed version of the interleaving distance for non-increasing functions from $\mathbb{R}_{\geq 0}$ to $\mathbb{R}_{\geq 0}$:

$$\begin{aligned} d_{\bowtie}(f,g) := \underset{\epsilon}{\text{minimize}} \quad & \int_0^\infty f(t) - g(t+\epsilon) + g(t) - f(t+\epsilon)\, dt \\ \text{s.t} \quad & f(t) \geq g(t+\epsilon)\ \forall t \in \mathbb{R}_{\geq 0} \\ & g(t) \geq f(t+\epsilon)\ \forall t \in \mathbb{R}_{\geq 0} \end{aligned}$$

One-parameter contour learning

Define a smoothed version of the feature counting invariant

The feature counting invariant is a step function, and we want to make it smooth.Define

$$\mathfrak{r}_\theta(F)(t) := \sum_{[a,b]\in B} \frac{1}{1 + e^{k(C_\theta(a,t)-b)}}$$

where B is the bar decomposition of F.

Choose the metric on persistence modules

Smoothed version of the interleaving distance for non-increasing functions from $\mathbb{R}_{\geq 0}$ to $\mathbb{R}_{\geq 0}$:

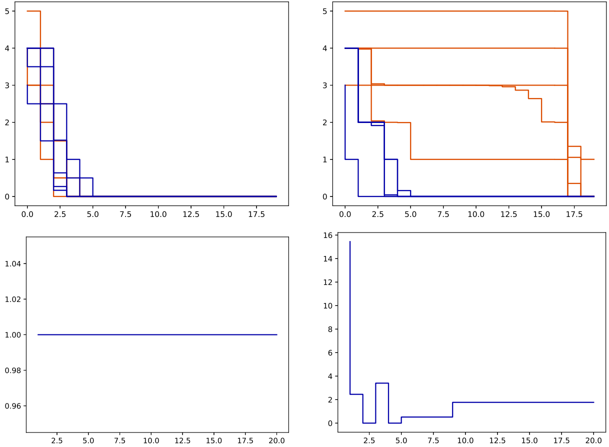
$$\begin{aligned} d_{\bowtie}(f,g) := \underset{\epsilon}{\text{minimize}} \quad & \int_0^\infty f(t) - g(t+\epsilon) + g(t) - f(t+\epsilon)\, dt \\ \text{s.t} \quad & f(t) \geq g(t+\epsilon)\ \forall t \in \mathbb{R}_{\geq 0} \\ & g(t) \geq f(t+\epsilon)\ \forall t \in \mathbb{R}_{\geq 0} \end{aligned}$$

Then,

$$d_\theta(F, F') := d_{\bowtie}(\mathfrak{r}_\theta(F), \mathfrak{r}_\theta(F'))$$

Example

- B consists of ten barcodes, with five in each class.
- Choose $\rho_\theta(\alpha) := \theta_i$ if $i \leq \alpha < i+1$ and we let $\theta_i = \theta_{i+1}$ if $i \geq 19$. Consequently, $\theta \in \mathbb{R}^{20}$.



Nonlinear Hierarchical Clustering

Nonlinear Hierarchical Clustering

For $n > 1$, the feature counting invariant is NP-hard to compute.

Nonlinear Hierarchical Clustering

For $n > 1$, the feature counting invariant is NP-hard to compute.

Theorem

If $F(v \leq w)$ is a surjection for all $v \leq w \in \mathbb{R}^n$, then:
$\widehat{\beta}_0(F)(\tau) = rank(F[\tau]) = dim(F(C(0, \tau)))$

Nonlinear Hierarchical Clustering

For $n > 1$, the feature counting invariant is NP-hard to compute.

Theorem

If $F(v \leq w)$ is a surjection for all $v \leq w \in \mathbb{R}^n$, then:
$\widehat{\beta}_0(F)(\tau) = rank(F[\tau]) = dim(F(C(0,\tau)))$

This happens for multiparameter clustering.

Nonlinear Hierarchical Clustering

For $n > 1$, the feature counting invariant is NP-hard to compute.

Theorem

If $F(v \leq w)$ is a surjection for all $v \leq w \in \mathbb{R}^n$, then:
$\widehat{\beta}_0(F)(\tau) = rank(F[\tau]) = dim(F(C(0, \tau)))$

This happens for multiparameter clustering.
Use the Hilbert function, $HF(v) := \dim F(v)$, to construct a smoothed feature counting function.

Nonlinear Hierarchical Clustering

For $n > 1$, the feature counting invariant is NP-hard to compute.

Theorem

If $F(v \leq w)$ is a surjection for all $v \leq w \in \mathbb{R}^n$, then:
$\widehat{\beta}_0(F)(\tau) = rank(F[\tau]) = dim(F(C(0,\tau)))$

This happens for multiparameter clustering.
Use the Hilbert function, $HF(v) := \dim F(v)$, to construct a smoothed feature counting function.
It can be decomposed into rectangular blocks of the same hight,

$$HF = \cup_i h_i I_{(a_{i1},\ldots,a_{in}),(b_{i1},\ldots,b_{in})}$$

Nonlinear Hierarchical Clustering

For $n > 1$, the feature counting invariant is NP-hard to compute.

Theorem

If $F(v \leq w)$ is a surjection for all $v \leq w \in \mathbb{R}^n$, then:
$\widehat{\beta}_0(F)(\tau) = rank(F[\tau]) = dim(F(C(0, \tau)))$

This happens for multiparameter clustering.
Use the Hilbert function, $HF(v) := \dim F(v)$, to construct a smoothed feature counting function.
It can be decomposed into rectangular blocks of the same hight,

$$HF = \cup_i h_i I_{(a_{i1},\ldots,a_{in}),(b_{i1},\ldots,b_{in})}$$

and thus

$$\mathfrak{r}_\theta(F)(\tau) = \sum_i h_i \prod_{k=1}^{n} \frac{1}{1 + e^{k(a_{ik} - C^i_\theta(0,\tau))}} \frac{1}{1 + e^{k(C^i_\theta(0,\tau) - b_{ik})}}$$

Nonlinear Hierarchical Clustering

- Solve problem (1) using gradient descent as in one-parameter case.

Nonlinear Hierarchical Clustering

- Solve problem (1) using gradient descent as in one-parameter case.
- Produces a nonlinear hierarchical clustering of the data by restricting the module to the curve outlined by $C(0, \tau)$.

Approximating the Feature Counting Invariant

Approximations

What problem are we approximating?

Approximations

What problem are we approximating?
Observation:

Theorem

$$\widehat{\beta}_0(F)(\epsilon) = min\{rank(G) \mid F[\epsilon] \subseteq G \subseteq F \text{ and } d(F, G) \leq \epsilon\}$$

Approximations

What problem are we approximating?
Observation:

Theorem

$$\widehat{\beta}_0(F)(\epsilon) = min\{rank(G) \mid F[\epsilon] \subseteq G \subseteq F \text{ and } d(F,G) \leq \epsilon\}$$

It suffices to minimize over the subfunctors of F.

Approximations

What problem are we approximating?

Observation:

Theorem

$$\widehat{\beta}_0(F)(\epsilon) = min\{rank(G) \mid F[\epsilon] \subseteq G \subseteq F \text{ and } d(F,G) \leq \epsilon\}$$

It suffices to minimize over the subfunctors of F.

$\{g_i \in F(v_i)\}$
Generating set for F

$\{F(v_i \leq C(v_i,\epsilon))(g_i) \in F(C(v_i,\epsilon))\}$
Genrating set for $F[\epsilon]$

Approximations

What problem are we approximating?

Observation:

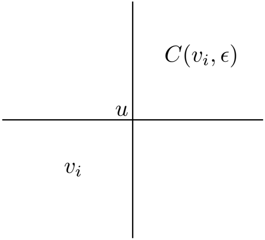
Theorem

$$\widehat{\beta}_0(F)(\epsilon) = min\{rank(G) \mid F[\epsilon] \subseteq G \subseteq F \text{ and } d(F,G) \leq \epsilon\}$$

It suffices to minimize over the subfunctors of F.

$\{g_i \in F(v_i)\}$	$\{F(v_i \leq C(v_i,\epsilon))(g_i) \in F(C(v_i,\epsilon))\}$
Generating set for F	Genrating set for $F[\epsilon]$

For ϵ large enough, there is u s.t:



Approximations

Want to find minimal number of elements in:

$$\bigcup_i F(u \leq v_i)^{-1}(F(v_i \leq C(v_i, \epsilon))(g_i)) \subseteq F(u)$$

that generate $F[\epsilon]$.

Approximations

Want to find minimal number of elements in:

$$\bigcup_i F(u \leq v_i)^{-1}(F(v_i \leq C(v_i, \epsilon))(g_i)) \subseteq F(u)$$

that generate $F[\epsilon]$.

Can be phrased as the following *matrix rank minimization problem with an affine constraint set*:

Input Vectors $x_1, \ldots, x_m \in \mathbb{R}^n$ and subspaces $L_1, \ldots, L_m \subseteq \mathbb{R}^n$
Output $\min\{\operatorname{rank}(\begin{bmatrix} c_1 & \ldots & c_m \end{bmatrix}) \mid c_i - x_i \in L_i\}$

Approximations

Want to find minimal number of elements in:

$$\bigcup_i F(u \leq v_i)^{-1}(F(v_i \leq C(v_i, \epsilon))(g_i)) \subseteq F(u)$$

that generate $F[\epsilon]$.

Can be phrased as the following *matrix rank minimization problem with an affine constraint set*:

Input Vectors $x_1, \ldots, x_m \in \mathbb{R}^n$ and subspaces $L_1, \ldots, L_m \subseteq \mathbb{R}^n$
Output $\min\{\text{rank}(\begin{bmatrix} c_1 & \ldots & c_m \end{bmatrix}) \mid c_i - x_i \in L_i\}$

This is NP-hard, by reduction from a graph colouring problem [2].

Example

Calculating $\widehat{\beta}_0(F)(2)$ for the following functor requires you to solve such a rank minimization problem:

5	0	0	0	0	0	0
4	K	K^2	K^3/L_2	0	0	0
3	K	K^2	K^3	K^3/L_1	0	0
2	K	K^2	K^3	K^3	K^3/L_0	0
1	0	K	K^2	K^2	K^2	0
0	0	0	K	K	K	0
	0	1	2	3	4	5

Approximations

$$\min_{A \in \mathcal{C} \subseteq \mathbb{R}^{m \times m}} \quad \operatorname{rank}(A)$$

Approximations

$$\min_{A \in \mathcal{C} \subseteq \mathbb{R}^{m \times m}} \quad \text{rank}(A)$$

Case $K = \mathbb{R}$: The problem can be efficiently approximated by the nuclear norm [3]:

Approximations

$$\min_{A \in \mathcal{C} \subseteq \mathbb{R}^{m \times m}} \quad \mathrm{rank}(A)$$

Case $K = \mathbb{R}$: The problem can be efficiently approximated by the nuclear norm [3]:

$$\min_{A \in \mathcal{C}} \quad \mathrm{rank}(A) \quad \sim \quad \min_{A \in \mathcal{C}} \quad ||A||_*$$

where $||A||_* = \sum \sigma_i(A)$.

Approximations

$$\min_{A \in \mathcal{C} \subseteq \mathbb{R}^{m \times m}} \quad \text{rank}(A)$$

Case $K = \mathbb{R}$**:** The problem can be efficiently approximated by the nuclear norm [3]:

$$\min_{A \in \mathcal{C}} \quad \text{rank}(A) \quad \sim \quad \min_{A \in \mathcal{C}} \quad ||A||_*$$

where $||A||_* = \sum \sigma_i(A)$.

Case $K = \mathbb{Z}/2\mathbb{Z}$**:** For each k, we can write the determinants of the $k \times k$ minors as Boolean clauses. Can be solved using a modern SAT solver, which can handle millions of variables and clauses.

Approximations

$$\min_{A \in \mathcal{C} \subseteq \mathbb{R}^{m \times m}} \quad \text{rank}(A)$$

Case $K = \mathbb{R}$: The problem can be efficiently approximated by the nuclear norm [3]:

$$\min_{A \in \mathcal{C}} \quad \text{rank}(A) \quad \sim \quad \min_{A \in \mathcal{C}} \quad ||A||_*$$

where $||A||_* = \sum \sigma_i(A)$.

Case $K = \mathbb{Z}/2\mathbb{Z}$: For each k, we can write the determinants of the $k \times k$ minors as Boolean clauses. Can be solved using a modern SAT solver, which can handle millions of variables and clauses.

Case $F = H_0(X(-); K)$: The F is a functor $F\colon \mathbb{R}^n \to \text{Sets}$ and the matrix rank minimization turns into a vertex covering problem (which can be approximated).

Next steps

Next steps

- Implementing and experimenting.

Next steps

- Implementing and experimenting.
- Generalizing contour learning to work for more classes of multi-persistence modules.

Next steps

- Implementing and experimenting.
- Generalizing contour learning to work for more classes of multi-persistence modules.
- Incorporating contour learning into common macine learning methods (logistic regression, SVM, etc).

References I

Peter Bubenik, Vin de Silva, and Jonathan Scott.
Metrics for generalized persistence modules.
Foundations of Computational Mathematics, 15(6):1501–1531, Dec 2015.

O. Gäfvert and W. Chachólski.
Stable Invariants for Multidimensional Persistence.
ArXiv e-prints, March 2017.

B. Recht, W. Xu, and B. Hassibi.
Necessary and sufficient conditions for success of the nuclear norm heuristic for rank minimization.
In *2008 47th IEEE Conference on Decision and Control*, pages 3065–3070, Dec 2008.

Martina Scolamiero, Wojciech Chachólski, Anders Lundman, Ryan Ramanujam, and Sebastian Öberg.
Multidimensional persistence and noise.
Foundations of Computational Mathematics, pages 1–40, 2016.