Algebraic Persistence — the algebra of persistence modules

Mikael Vejdemo-Johansson
joint with Primoz Skraba

School of Computer Science
University of St Andrews
Scotland

4 June 2012

Outline

Persistence modules

Introduced and identified by Zomorodian and Carlsson (2005).

Definition

A persistence module M is a graded module over the graded ring $\Bbbk[t]$.

Connection to persistent homology

Filtered chain complexes and their persistent homology both are persistence modules.

A filtered chain complex has a generator in degree n for each simplex appearing at filtration step n.

A persistent homology module has a generator in degree n killed by t^m for each barcode entry $(n, n+m)$.

Category of persistence modules

Thus, to study persistent homology, we will benefit from studying the category of persistence modules – which by the results by Zomorodian and Carlsson means studying the category of graded modules over $\Bbbk[t]$.

Very nice ring. Very nice category. Here are some things that are true:

- Euclidean domain. Division algorithm works. Also, therefore PID.
- Submodules of free modules are free. All modules have a presentation by a short exact sequence $0 \to R \to G \to M \to 0$ where R, G are both *free* modules.

Outline

Nested modules

Since persistence modules have canonically free presentation, we can represent a persistence module by tracking the generators and relations. There are two ways to do this with a global module C of chains:

Represent chains

We maintain matrices representing

$$G \to C \text{ and } R \to C$$

- \+ This is the output from existing algorithms
- \+ We can work with each matrix separately
- \- Larger matrices

Represent relations embedding

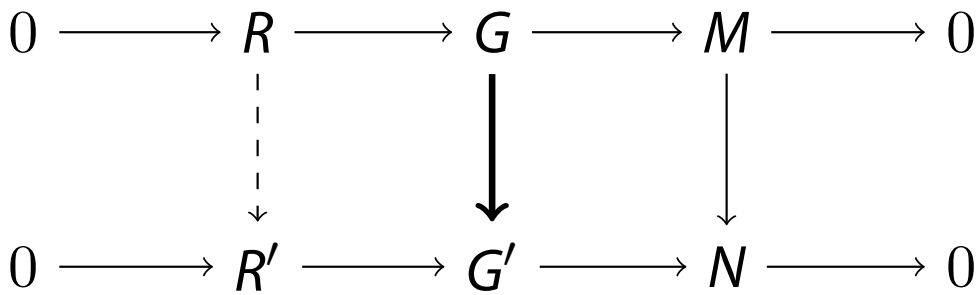
We maintain matrices representing

$$G \to C \text{ and } R \to G$$

- \+ We have swifter access to the barcode
- \- We have to modify both matrices simultaneously

Homomorphisms as matrices with conditions

A homomorphism between two modules can be represented by images of the generators such that boundaries all map to boundaries.



To represent a homomorphism $M \to N$, it is enough to work with a homomorphism $G \to G'$ known to map relations to relations.

This corresponds to the well-formed map requirement in **Cohen-Steiner, Edelsbrunner, Harer, Morozov**: *Persistent Homology for Kernels, Images, and Cokernels.*

Outline

Normal forms, equality, and membership

Question

How can we determine equality for two elements of $M = G/R$?

Question

How can we determine whether $z \in C$ represents an element of $M = G/R$?

Question

How can we determine whether $z \in G$ represents an element of R?

Question

How do we produce bases for G and R that make computation easy?

Normal forms, equality, and membership

Answer

A **Gröbner basis** comes with extensive computational guarantees.

Reduction modulo a Gröbner basis, in any order, until no more pivots (leading elements) apply is guaranteed to provide a normal form. Normal form equal to 0 implies membership. Equal normal form implies equality (modulo the Gröbner basis).

For modules over a field $\Bbbk$, a Gröbner basis is equivalent to a reduced echelon form (REF).

Helpful fact

The ring $\Bbbk[t]$ is sufficiently much like a field – a Gröbner basis of graded modules is also equivalent to a reduced echelon form.

Normal forms, equality, and membership

We shall want to maintain G and R with bases and normal form such that R is always represented by a REF, and G is always reduced with respect to R.

We can avoid redundancy by keeping a basis for G reduced to a REF as well.

This is in particular important since the persistence algorithm itself works with a membership test in the relations module as the fundamental step.

Graded Smith normal form

There is a way to compute a Smith Normal Form in a graded sense.

Properties of a Graded Smith Normal Form

- Rows are ordered by increasing degree
- Columns are ordered by increasing degree
- Each row has at most one non-zero entry
- Each column has at most one non-zero entry
- Lower degree entries divide all higher degree entries

Strictly speaking, this is a permutation of the classical Smith Normal Form.

Graded Smith normal form

Core feature:

Computability

We can compute a Graded Smith Normal Form by reducing rows and columns in increasing order of degree. Thus we can compute it compatibly with the gradings present.

Conditions

To do this, we require the coefficients to come from a graded principal ideal domain. $\Bbbk[t]$ fulfills this requirement.

SNF and barcodes

Why should we care about Smith normal forms?

Persistence modules and barcodes

A graded Smith normal form of the inclusion map $R \to G$ is the same thing as a barcode of $M = G/R$.

Proof sketch

A graded Smith normal form is a simultaneous basis choice of R and G such that each basis element of R maps onto a $\Bbbk[t]$-multiple of a basis element of G.
This is exactly what produces a barcode: bases for cycles and boundaries such that each boundary basis element kills exactly one cycle basis element.

Free kernels and intersections

One technique that will show up a lot in the subsequent constructions is to compute a kernel of a map between free modules. This is done using a REF computation:

- Reduce the matrix of the map to a REF, tracking the operations performed.
- Operation combinations corresponding to 0-rows are generators of the kernel.

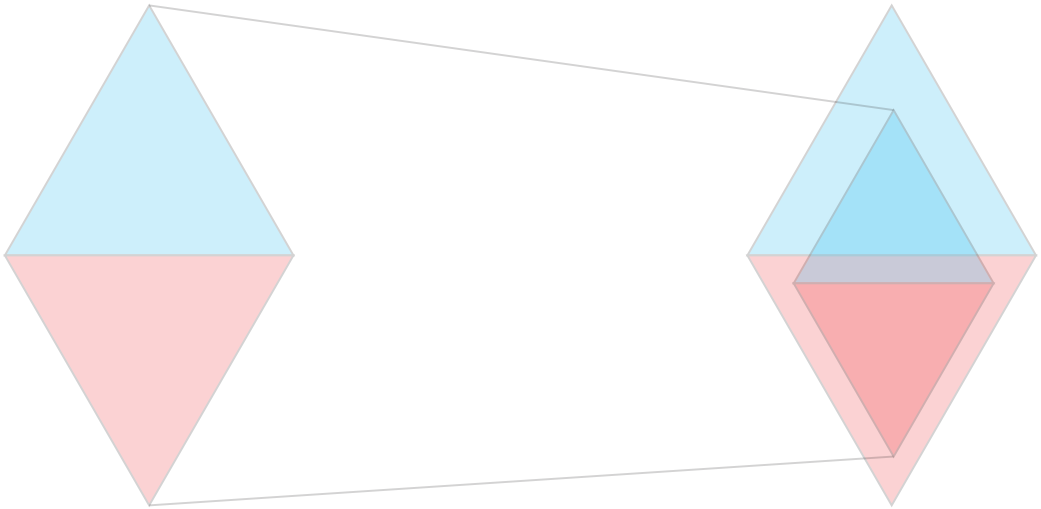
This can be used to compute $M \cap N$ where M, N are free submodules of some free module C. The map $M \oplus N \to C$ given by $(m, n) \mapsto m - n$ has a kernel consisting of pairs (m, n) such that $m = n$ in C. Projecting onto one factor gives a basis for the intersection.

Illustration

$M = G/R$ and $N = G'/R'$.

Illustration

$f : M \to N$ represented by $\phi : G \to G'$ such that $\phi(R) \subset R'$.



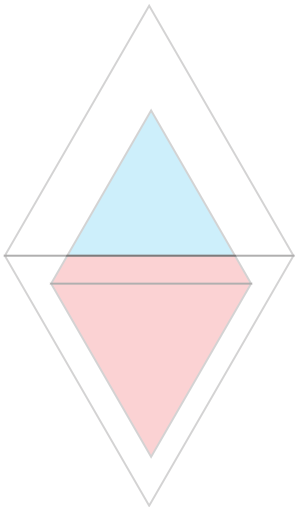
We shall be illustrating the various constructions based on this figure.

Image

$f : M \to N$ represented by $\phi : G \to G'$ such that $\phi(R) \subset R'$.

- Compute $\phi(g)$ for each basis element $g \in G$.
- Reduce images modulo the REF for R'.
  - These are the generators for im f.

For the relations, we need to compute a basis for $G' \cap \phi(R)$. This is done with another REF computation.

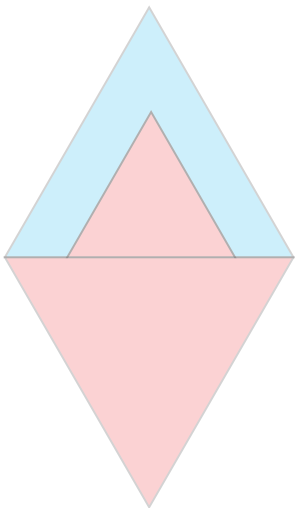


Cokernel

$f : M \to N$ represented by $\phi : G \to G'$ such that $\phi(R) \subset R'$.

- Compute $\phi(g)$ for each basis element $g \in G$.
- Reduce the basis of G' by the images $\phi(g)$.

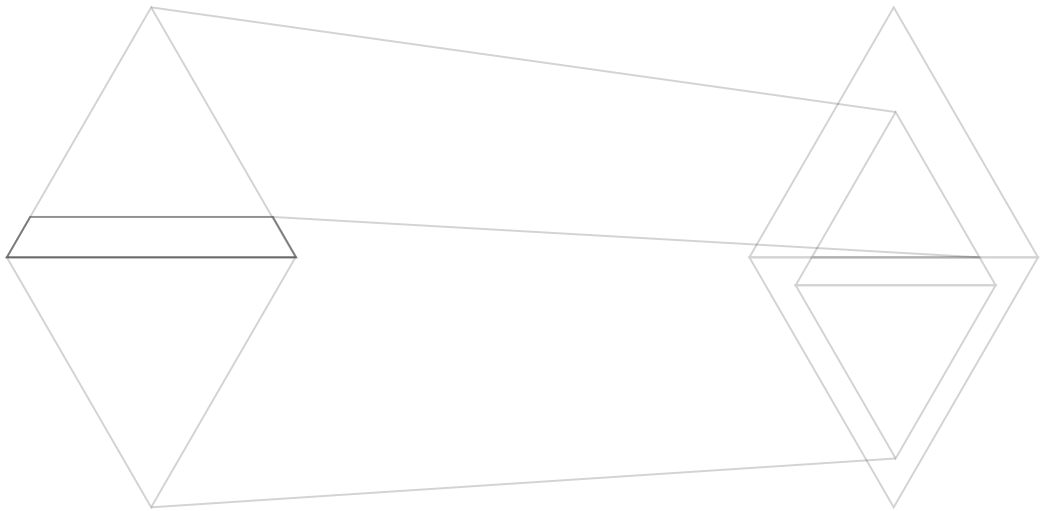
This gives you generators for coker f. The relations are those in R together with all the images $\phi(g)$.



Kernel

$f : M \to N$ represented by $\phi : G \to G'$ such that $\phi(R) \subset R'$.

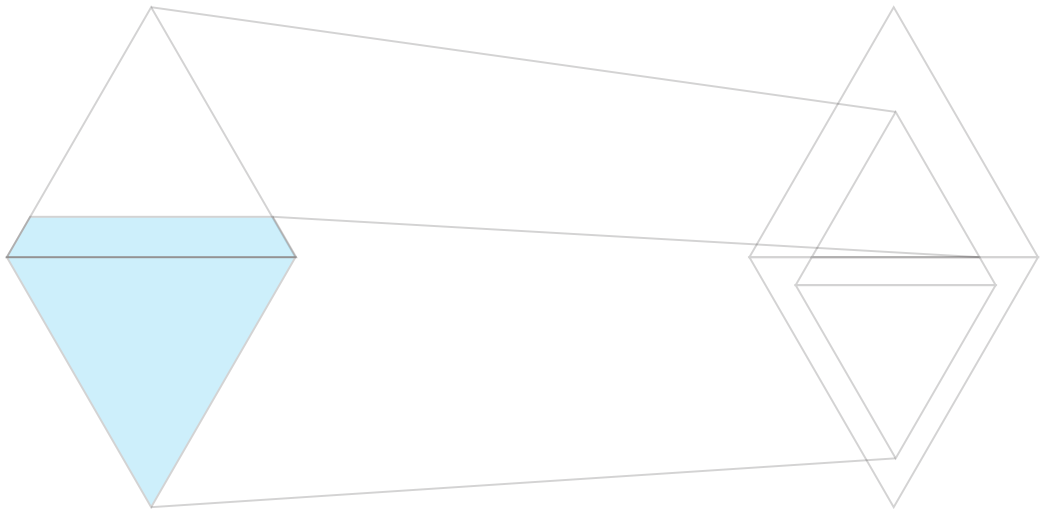
Computing the kernel is a two-step process. One step computes the generators, and the next step computes the relations.



Kernel (Step 1: Generators)

$f : M \to N$ represented by $\phi : G \to G'$ such that $\phi(R) \subset R'$.

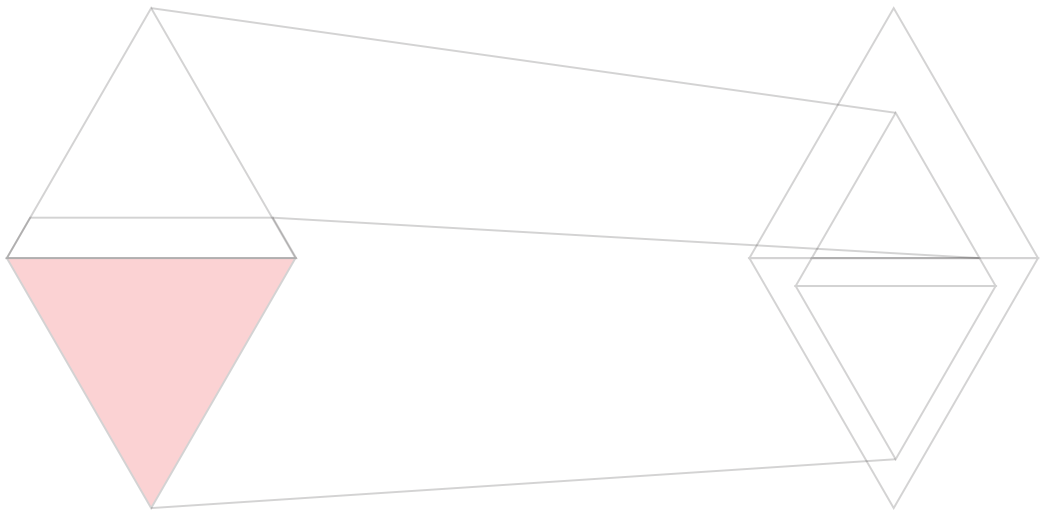
Generators are the basis of a kernel of the map $G \oplus R' \to G'$ given by $(g, r) \mapsto \phi(g) - r$. Project onto the first factor to get an embedding into G. Call these kernel generators K, and the projection $i : K \to G$.



Kernel (Step 2: Relations)

$f : M \to N$ represented by $\phi : G \to G'$ such that $\phi(R) \subset R'$.

Relations are the basis of a kernel of the map $K \oplus R \to G$ given by $(k, r) \mapsto i(k) - r$. The projection onto K is the inclusion map of relations into generators for the kernel.



Pushouts and pullbacks

If you know enough category theory, you'll recognize this $\ker(A \oplus B \to C)$ pattern as being a pullback. These descriptions above provide all tools necessary to compute both pullbacks and pushouts, even for torsion persistence modules:

Pullback Given $f : A \to C$ and $g : B \to C$, the pullback is $\ker((a, b) \mapsto f(a) - g(b))$.

Pushout Given $f : A \to B$ and $g : A \to C$, the pushout is $\operatorname{coker}(a \mapsto (f(a), g(a))$.

Outline

Torsion chain complexes

One perennial problem in persistence is how to handle *torsion* on a chain level; what if simplices can disappear again?

First solution

Zig-zag homology has provided one solution: vanishing simplices are modelled with inclusions going *the other way*. (de Silva, Morozov, Carlsson)

Our approach

Torsion in the chain complex can be modelled by allowing non-trivial relations in the chain complex.

We note that these approaches lead to different results. In particular, our approach models *relative homology*.

Relative (co)homology

Classically

$$H_*(X;A) = H_*(C_*X/C_*A)$$

Our approach to modeling non-free persistence modules gives us all the tools necessary to work with a chain complex like C_*X/C_*A.

In particular, since ∂ is a map of persistence modules $C_*X/C_*A \to C_*X/C_*A$, we can compute coker ker $\partial = H_*(X;A)$.

Unordered input

Inspired by this view-point, we can adapt the classical persistence algorithm to one that will not require ordered input.

Algorithm: out of order persistence

For each simplex σ:

1. Compute $d\sigma$.
2. Reduce $d\sigma$ modulo all earlier boundaries
3. If $d\sigma$ reduces to 0, then σ starts a new cycle. Loop.
4. Otherwise, $d\sigma$ is a new boundary. Find latest simplex τ in reduced $d\sigma$ and construct the pair (τ, σ). If τ was already in a pair (τ, ψ), reduce $d\psi$ modulo σ and continue the algorithm for ψ.