VII 2. Efficient Implementations

Sparse Matrix Representation

Recall that we used the matrix reduction of ∂ to compute the persistence homology of the filtration $K_0 \subseteq K_1 \subseteq \cdots \subseteq K_n = K$, where $K_i = \{\sigma_0, \ldots, \sigma_i\}$. The resulting reduced matrix R consists of the information of the persistence diagram. (See VII 1)

Since the columns of ∂ consist of simplices of all dimension and the boundary of a simplex is a codimension-1 chain, the matrix ∂ is sparse (by convention, a matrix is "sparse" if its number of nonzero entries is approximately the number of columns / rows).

In most cases, R is also sparse. There are more efficient ways to store sparse matrices.

① Dictionary of keys (DOK): Store a sparse matrix by collections of dictionaries that map (row, column)-tuples to the values. In the case of 0-1 matrices, only (row, column)-tuples. This method is convenient for construction but poor for iteration.

② List of list (LIL): Store one list per column/row, with each list consists of tuples of (row/column index, value)-pairs. In the case of 0-1 matrices, only 0-1 matrices. Typically, the inner lists are sorted by increasing/decreasing row/column indices.

Matrix Reduction using sparse matrix representation

The boundary matrix $\partial_{n \times n}$ is given.

Step 0: Store the boundary matrix by LIL, one list per column, with row indices sorted decreasingly (the most recently added simplex is at the top). Denote each list $\tilde{\partial}[j]$. $\tilde{\partial}[j] = [\text{low}(j), \ldots]$

⋮

Step j: Create an empty list $\tilde{R}[j]$. Put the j^{th} column of the reduced matrix R on $\tilde{R}[i]$, where $i = \text{low}(j)$. If $\text{low}(j)$ undefined, do nothing. We do this by the loop

$L = \tilde{\partial}[j]$ ($L[0]$ is the index of the lowest 1 of L)

while $L \neq$ null. && $\tilde{R}[L[0]] \neq$ null. do

$L = L + R[L[0]]$ By "+" we mean merge the two lists and discard duplicate terms.

endwhile

If $L \neq$ Null. then set $\tilde{R}[L[0]] = L$. endif.

The condition "$L \neq$ null" means L is not an 0 after previous reduction. Because of the last command. "$\tilde{R}[L[0]] \neq$ null" means there is no previous reduced column L' with $L'[0] = L[0]$. i.e. $low(L') = low(L)$.

If L is empty after some addition, then the j-th iteration is end.

If the "if" is executed, then the "while" loop is end. i.e. $\tilde{R}[L[0]] =$ null is avaliable. we set its value to be L.

As a result. the i-th array $\tilde{R}[i]$ of $\tilde{R}$ stores the j-th column of the reduced matrix R. where $i = low(j)$. If such j do not exist. $\tilde{R}[i] =$ null.

What's good: ① There are fast algorithms for sparse matrices.

② At the j-th iteration. we need not to compare all $low(j')$s with $low(j)$ for $j' < j$. It suffices to check the emptiness of $R[L[0]]$.

Time complexity: The time complexity of the j-th iteration is $(p+1)(j-i)^2$. where p is the dimension of σ_j. Indeed. in the while loop. $L(0)$ is strictly decreasing ($low(j)$ is strictly increasing). until it reach $i = low(j)$. Thus the largest number of addition is $j-i$. (we try to store L in $\tilde{R}[k]$ for some $i \leq k < j$ every time. If it fails. an addition occur.) At each addition. the lists added contain only nonzero terms at faces of simplices with index in $[\![i, j]\!]$. There are at most $(j-i)(p+1)$ such simplices. Therefore. the time complexity is $(j-i)^2(p+1)$. In practice it is usually much faster than this.

Thm (Persistence equivalence theorem) Let $\phi_i : U_i \to V_i$ be isomorphism of vector spaces. If the diagram

$$\begin{array}{ccccccc} V_0 & \longrightarrow & V_1 & \longrightarrow & \cdots & \longrightarrow & V_n \\ \uparrow & & \uparrow & & & & \uparrow \\ U_0 & \longrightarrow & U_1 & \longrightarrow & \cdots & \longrightarrow & U_n \end{array}$$

commute. then

the persistence diagram of the $\{v_i\}$'s are the same as the $\{u_i\}$'s.

Pf: Omitted.

Rmk: In practice. $V_i = H_p(K_i)$ for some filtration K_i. Φ_i are isomorphisms commuting with $f_\alpha^{i,j}: H_p(K_i) \to H_p(K_j)$. A sequence of such isomorphisms gives the equivalence of "persistence vector space", that is, a seq of vector space $\{V_i\}$ with linear maps $f^{i,j}: V_i \to V_j$ s.t. the composition law holds.

Surfaces

Let K be a simplicial complex that triangulates a 2-manifold.

Recall that we can define a piecewise linear function $f: |K| \to \mathbb{R}$ by specifying the values on the vertices $f(u_1) < f(u_2) < \cdots < f(u_n)$ and extend to the whole $|K|$ using barycentric coordinates. $f(x) = \sum_i b_i(x) f(u_i)$.

Then we obtain the lower star filtration of f. $K_0 \subseteq K_1 \subseteq \cdots \subseteq K_n = K$,

where $K_i = \bigcup_{f(v_j) \le f(u_i)} \overline{St}(u_i)$. If we define $g: K \to \mathbb{R}$ by $g(\sigma) = \max_{x \in \sigma} f(x)$, then for $f(u_i) \le a < f(u_{i+1})$, we have $K_i = g^{-1}(-\infty, a]$. Let $K_a := g^{-1}(-\infty, a]$

As discussed in VI.3. $|K_i|$ is a deformation retract to $f^{-1}(-\infty, a]$;

Also observe that $|K_a| \subseteq |K|_a := f^{-1}(-\infty, a]$ for an $a \in \mathbb{R}$.

Hence the map $H_p(K_a) \xrightarrow{\sim} H_p(|K|_a)$ induced by inclusion is an isomorphism.

The commuting diagram $H_p(|K|_a) \to H_p(|K|_b)$, $a \le b$ together with

$$\begin{array}{ccc} H_p(|K|_a) & \to & H_p(|K|_b) \\ \uparrow & & \uparrow \\ H_p(K_a) & \to & H_p(K_b) \end{array}$$

the persistence equivalence theorem implies that calculating the persistence homology of the lower star filtration of K is equivalent to that of f.

Zeroth Diagram

There is a much more efficient method to computing the 0-th diagram. Given a 1-simplex σ. Suppose $\partial\sigma = u + v$ and $u < v$. If u and v are already in the same connected component, then the addition of σ does not affect the zeroth diagram. Otherwise, it kills a 0-cycle that is

created by the younger one of the oldest simplex of the two connected components.

Thus all we need to know is whether u and v belong to different connected components, and if they do, the oldest simplices of each component. This is exactly the Kruskal's algorithm for disjoint set systems we introduced in chapter I.1. A simplicial complex with only 0,1 simplices is just a simple graph!

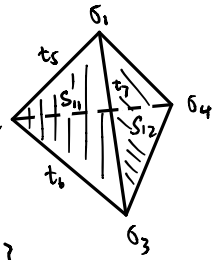
Recall in section I.1. when u, v are not in the same connected component, we point the smaller component in the larger one. However in this case, we point the younger component to the older one because the younger component is the one being killed.

This algorithm has time complexity $O(n\alpha(n))$ after doing parallization. where α is the inverse of the Ackermann function. which is a constant in practice.

First diagram

If K is a triangulation of a 2-manifold (without boundary). $f: K \to \mathbb{R}$ be an increasing function (i.e. $\sigma \le \tau \Rightarrow f(\sigma) \le f(\tau)$), we consider $-f: K \to \mathbb{R}$. For $a \in \mathbb{R}$. $(-f)^{-1}(-\infty, -a] = K - f^{-1}(-\infty, a] = (f^{-1}(-\infty, a))^c$. Let $\widetilde{K}_i = (-f)^{-1}(-\infty, -a_i]$. we obtain $\widetilde{K}_0 = K \supseteq \widetilde{K}_1 \supseteq \cdots \supseteq \widetilde{K}_n = \phi$. When adding a simplex σ to K_i. $K_{i+1} = K_i \cup \{\sigma\}$. σ is deleted from $\widetilde{K}_i$ and $\widetilde{K}_i = \widetilde{K}_{i+1} \cup \{\sigma\}$.

Each $\widetilde{K}_i$ is not a simplicial complex. but we can still consider the (singular) homology of its underlying space. that is $|K| - |K_i| \subseteq \mathbb{R}^n$.



E.g. $K = \{\sigma_1, \ldots, \sigma_4, t_5 = (\sigma_1\sigma_2), t_6 = (\sigma_2\sigma_3), t_7 = (\sigma_1\sigma_3), \ldots, t_{10}, s_{11} = (\sigma_1\sigma_2\sigma_3), \ldots s_{14}\}$. a triangulation of S^2. The point (7.11) is on the 1-st persistence diagram of the filtration $K_0 \subseteq \cdots \subseteq K_n$. I.e. the cycle $t_5 + t_6 + t_7$ is created at K_7. and dies at K_{11}. Now consider $\widetilde{K}_{11} = \{s_{11}, s_{12}, s_{13}, s_{14}\}$. having 4 disjoint 2-simplices, which we view as connected components. s_{11} is the one appear with the addition of s_{11}. s_{11} stays isolated until

the addition of t_7 in $\widetilde{K}_7 = \{t_7, \dots, t_{10}, s_{11}, \dots, s_{14}\}$. which connect S_{11} to S_{12}. Thus a zero cycle in $\widetilde{K}_n \subseteq \widetilde{K}_{n-1} \subseteq \cdots \subseteq \widetilde{K}_0$ is created by S_{11} and killed by t_7. In other words. $(-11.-7) \in Dgm_0(-f)$ $(\widetilde{K}_i = (-f)^{-1}(-\infty.-a_i))$

In general. $(a.b) \in Dgm_1(f) \Leftrightarrow (-b.-a) \in Dgm_0(-f)$. The rigorous construction and proof will be treated next chapter.

Therefore. the computation of $Dgm_1(f)$ is reduced to that of $Dgm_0(-f)$, which can be done by Kruskal's algorithm whose time complexity is $O(n) \simeq$ Const.