SM 402: Basic Computational Topology May 13, 2023

Research Paper Summary

Madhav Sood (IMT2021009)
Vidhish Trivedi (IMT2021055) *Instructor:* Prof. Amit Chattopadhyay

Summary of the paper:
Topological Persistence and Simplification - Herbert Edelsbrunner • David Letscher • Afra Zomorodian

Overview:

The paper introduces several key contributions to the field of computational topology and data analysis.

Firstly, the paper defines the concept of persistence for Betti numbers and nonbounding cycles, which provides a measure of the significance and lifetime of topological features in a data set.

Secondly, the paper presents an efficient algorithm for computing persistence, which is applicable to a wide range of data types, including point clouds, digital images, and biomedical data. The algorithm discusses the persistence diagram, which visually represents the evolution of topological features over a range of parameter values.

Finally, the paper describes a simplification algorithm based on persistence, which can reduce the complexity of a data set by identifying and removing topological features with low persistence. This approach allows for the extraction of the essential structure of the data while discarding irrelevant details, which can be useful for applications such as data visualization and pattern recognition.

Important Concepts in the paper:

1. Chains, cycles and Boundaries:

In the context of a simplicial complex K in $\mathbb{R}^3$, a k-chain is defined as a subset of k-simplices in K. The set of all k-chains, along with the addition operation, forms a group called C_k. The boundary operator $\delta_k : C_k \rightarrow C_{k-1}$ is defined as the collection of all (k-1)-dimensional faces of a k-simplex σ. Each boundary operator is a homeomorphism.

The kernel of δ_k is the collection of k-chains with empty boundary, which is also known as k-cycles. The image of δ_k is the collection of (k-1)-chains that are boundaries of k-chains. A k-boundary is a k-chain in the image of δ_{k+1}. Both k-cycles and k-boundaries denoted as Z_k and B_k respectively are defined as subgroups of C_k with the addition operation.

2. Homology Groups:

The k-th homology group is the k-th cycle group factored by the k-th boundary group, $H_k = Z_k/B_k$.
The k-th Betti number of K is the rank of the k-th homology group, $\beta_k = rank\ H_k$.
As $H_k = Z_k/B_k$, $rank H_k = rank Z_k - rank B_k$.

3. Age Filters:

The paper introduces a method for computing topological features of a simplicial complex based on an ordering of its simplices. This ordering is called a filter, and it has the property that each prefix of the ordering contains the simplices of a subcomplex. By taking successively larger prefixes of the filter, a sequence of subcomplexes is defined, called filtration.

4. Positive and Negative Simplex:

We call a (k + 1)-simplex σ^i positive if it belongs to a (k + 1)- cycle and negative otherwise.
In other words, σ^i is a positive simplex if it creates a cycle when it enters with respect to a filtration, and negative if it destroys a cycle when it enters with respect to a filtration.

5. Non-bounding Cycles:

A non-bounding cycle is a cycle in a simplicial complex that cannot be expressed as the boundary of any chain. In other words, it is a closed loop that does not "wrap around" any hole or void in the complex

6. Persistence:

Persistence is a measure of the significance and lifetime of topological features in a dataset. We define Z_k^l, B_k^l to be the k-th cycle group and k-th boundary group, respectively, of the l-th complex K^l in a filtration. To capture persistent cycles in K^l, we factor its k-th cycle group by the k-th boundary group of K^{l+p}, p complexes later in the filtration. Formally, the p-persistent k-th homology group of K^l is: $H_k^{l,p} = Z_k^l/(B_k^{l+p} \cap Z_k^l)$. The p-persistent k-th Betti number β_k^{l+p} of K^l is the rank of H_k^{l+p}.
In persistence homology, a cycle is said to persist for p steps if it appears as a boundary of a higher-dimensional simplex in the simplicial complex at step k, but is not itself a boundary of any simplex until step k + p.
The p-persistent homology groups can also be defined using injective homomorphisms between ordinary homology groups.

Pairing Algorithm:
The main algorithm given in the paper is the pairing algorithm which helps us find the persistence of cycles in a given filtration and hence find the persistence homology groups. The algorithm pairs up each positive simplex with its corresponding negative simplex that is responsible for destroying the cycle that is created by the positive simplex. The algorithm is as follows:

1. Initialize an empty basis H_k for the k-th homology group.
2. For each positive k-simplex σ^i, do the following:
   - Find a non-bounding k-cycle c^i that contains σ^i but no other positive k-simplices.
   - Add the homology class of c^i as a new element to the basis of the k-th homology group H_k.
3. For each negative (k+1)-simplex σ^j, do the following:
   - Find its corresponding positive k-simplex σ^i and remove the homology class of σ^i from the basis.
   - Identify σ^j as the simplex responsible for turning the k-cycle created by σ^i into a k-boundary.
   - Append (σ^i, σ^j) to the list L_k.
   - Compute the persistence of the k-cycle, which is one less than the difference between the indices (in filtration) of the two simplices i.e $j - i - 1$ or the difference in their birth times. (Birth times is the time at which they were added in the filtration.)

Algorithm 1 list Pair-Simplices()

```
1: $L_0 = L_1 = L_2 = \phi$
2: for $j = 0$ to $m - 1$ do
3:     $k = dim\ \sigma^j - 1$
4:     if $\sigma^j$ is negative then
5:         $d = \delta_{k+1}(\sigma^j)$;
6:         $i = y(d)$;
7:         $L_k = L_k \cup \{(\sigma^i, \sigma^j)\}$;
8:     end if
9: end for
10: return ($L_0 = L_1 = L_2$)
```

The pairing algorithm gives us a set of simplex pairs (σ^i, σ^j), each representing a k-cycle for $0 \leq$ k ≤ 2.
The collection of positive k-simplices $\Gamma = \Gamma($ d $)$ is uniquely determined by d. The youngest simplex in Γ is the one with the largest index and we denote this index as y(d).

To compute the index of the positive simplex to be paired up with each negative simplex in the above algorithm, a **cycle search** algorithm is used. This is done by computing the index i of the youngest positive k-simplex in $\Gamma(d)$, where d = $\delta_{k+1}(\sigma^j)$.
Therefore, steps 5 and 6 in the above pseudo-code are performed by the cycle search algorithm.

Cycle Search Algorithm:

Data Structure Used:
The algorithm uses a linear array T[0..m-1] as a hash table to store pairs identified by the algorithm. Each

simplex in the filter has a slot in the hash table, but information is stored only in the slots of the positive simplices. The information stored at the $T[i]$, where i is the index of a positive simplex, consists of the index j of the matching negative simplex and a list of positive simplices defining a cycle. Some cycles exist beyond the end of the filter, in which case ∞ is used as a substitute for j.

The Cycle Search algorithm is as follows:

1. Given the index j and a negative (k+1)-simplex σ^j, identify the set $\Gamma(d)$ of positive k-simplices that represent the homology class of d, the boundary of σ^j in H_k^{j-1}.
2. Initialize a set A as the set of positive k-simplices in d, and set i = $\max(A)$, the index of the youngest member of A.
3. Probe from right to left in the data structure for the slot T[i].
4. If T[i] is unoccupied, store j and A in T[i] and end the search.
5. If T[i] is occupied, it contains a collection A^i representing a permanently stored k-cycle that is already a k-boundary. In this case a collision occurs. We add A and A^i (take the symmetric difference of the two) to get a new A representing a k-cycle homologous to the old one and therefore also homologous to d. Update i as the index of the youngest member of the new A and repeat the above steps until an unoccupied slot is found.

Algorithm 2 integer YOUNGEST (simplex σ^j)

```
1: $A = \{\sigma \in \delta_{k+1}(\sigma^j) \mid \sigma \textit{ positive}\}$
2: loop
3:     i = max($A$)
4:     if T[$i$] is unoccupied then
5:         store j and A in T[i]
6:         exit the loop
7:     end if
8:     $A = A + A^i$
9: end loop
```

Running time of Cycle Search:

Let d = $\delta_{k+1}(\sigma^j)$ and let σ^i be the youngest positive k-simplex in Γ(d). The persistence of the cycle created by σ^i and destroyed by σ^j is p_i = j - i - 1. The search for σ^i proceeds from right to left starting at T[j] and ending at T[i]. The number of collisions is at most the number of positive k-simplices strictly between and σ^i and σ^j, which is less than pi. k-cycle defined by A^i is the sum of fewer than p_i boundaries of (k + 1)-simplices.
A single collision takes time at most $O(p_i)$, and the entire search for σ^i takes time at most $O(p_i^2)$.

$\therefore$ Running time of cycle search = $O(p_i^2)$.

Running time of Pairing Algorithm:

From above, Running time of cycle search = $O(p_i^2)$. The total algorithm runs in time at most $O(\sum p_i^2)$.

$\therefore$ Running time of Pairing Algorithm is $O(m^3)$.

Problem with the Pairing Algorithm:

The problem with the pairing algorithm is that there is no way, given in the paper, to compute whether a given simplex is positive or negative in a given filtration. Thus we move on to the next algorithm (as discussed with the professor) to find the persistence. The following is also the algorithm implemented by us in our code.

Matrix Reduction Algorithm:

This algorithm was originally described in the book "Computational Topology: An Introduction" authored by Herbert Edelsbrunner and John Harer.
Let K be a simplicial complex, and $\sigma^1 < \sigma^2 < \ldots\ldots < \sigma^n$ an ordering of its simplices corresponding to its filtration. Define the boundary matrix of K, denoted by δ as follows:

δ is a $n \times n$ matrix where -

$$\delta(i,j) = \begin{cases} 1, & \sigma^i \text{ is a co-dimension one face of } \sigma^j \\ 0, & \text{otherwise} \end{cases}$$

The ordering of rows and columns follows the total ordering of the simplices. The column corresponding to a simplex records its boundary. The algorithm uses column operations to reduce δ to another 0-1 matrix R. Let low(j) be the row index of the lowest one in column j. If the entire column is zero then $low(j)$ is undefined. We call R reduced if $low(j) \neq low(j_0)$ whenever $j \neq j_0$ specify two non-zero columns. The algorithm reduces δ by adding columns from left to right. The following pseudo-code allows us to compute the reduced matrix:

Algorithm 3 Reduction of the Boundary Matrix

```
1: $R = \delta$
2: for j = 1 to n do
3:     while there exists $j_0 < j$ with $low(j_0) = low(j)$ do
4:         add column $j_0$ to column $j$
5:     end while
6: end for
```

The process of reducing columns to zero is called Gauss reduction.
The **running time** of the above algorithm is at most cubic in the number of simplices.

In the reduced matrix, if $low(j)$ is undefined, then the simplex σ^j is a positive simplex, otherwise it is a negative simplex.

For pairing,

(σ^i, σ^j) will form a persistence pair iff $i = low(j)$. (i corresponds to a row and j to a column).
Here, if σ^i is a simplex of dimension k, then σ^j is a simplex of dimension (k+1).

Also, we have (σ^i, ∞) as a persistence pair iff column i is zero but row i does not contain a lowest one.

Visualisation:

The k-interval of a pair of simplices (σ^i, σ^j) is extended into a k-triangle in the index-persistence plane. The k-triangle is defined by three vertices: (i, 0), (j, 0), and (i, j-i), where i and j are indices representing the birth and death times of the simplices, respectively.
The k-triangle is closed along its vertical and horizontal edges, which correspond to the persistence intervals of the simplices, and open along the diagonal connecting (j, 0) to (i, j-i). This diagonal represents the persistence interval of the k-cycle created by σ^i, which is progressively destroyed by σ^j as we increase the threshold value p. $\beta_k^{l,p}$ is the number of k-triangles that contain point (l,p).

Simplification:

The paper proposes a method for simplifying a topological persistence diagram by identifying pairs of topological features that can be merged through migration.

Migration: It involves reordering the filtration to obtain a new filtration whose Betti numbers are the same as the p-persistent Betti numbers of the original filtration.

The paper discusses the reordering of a filtration to obtain a simplified version of the original filtration with valid k-intervals. This is done by moving σ^j closer to or all the way to σ^i for each pair (σ^i, σ^j) and determining the new position of σ^j. The new position of σ^j is max{i, j-p}

A complication arises when a negative simplex attempts to move past one of its faces, requiring the face to move along with its coface to maintain the ordering as a filter. The algorithm recursively moves all necessary faces and their matching negative simplices for any moving simplex.
However, conflicts arise when the face represents a cycle whose persistence is at least p, which can occur due to the structure of a complex and the recursive nature of any simplification algorithm.

Conclusion:

The paper introduces the notion of topological persistence for a filtration in R^3 and gives algorithms for assessing persistence and simplifying the filtration.