OPEN ACCESS Freely available online

Journal of Proteomics & Bioinformatics

Research Article

A Mathematical Model for Protein Structure Prediction and Analysis

Zakaria Lamine[1,2*], Mohammed Wadia Mansouri[1], My Ismail Mamouni[2]

[1]Department of Mathematics, Ibnou Tofail University, Kenitra, Morocco; [2]Department of Mathematics, CRMEF, Rabat, Morocco

ABSTRACT

In the aim of presenting a learning approach derived from algebraic topology for protein structure prediction, we will be showing how our quotient spaces could qualitatively give insight into how building good homomorphism's can help identifying accurate neural networks. We will also be giving as an example of application the use of a model generated after extracting an algebraic invariant which is in our case a persistent diagram on some biological data, by encoding the two first homologies H_1 to H_0 using a boundary operator, the algorithms are originated from algebraic geometry. Basically two main algorithms are used the Buchberger's algorithm and Shreyer's algorithm.

Keywords: Neural networks; Persistent diagrams; Buchberger's algorithm; Shreyer's algorithm

INTRODUCTION

The main idea of this paper is reconstructing molecular shapes by using alternative to the interpreted graph neural networks that are using geometric parametres for building artificial intelligence models, so we can access a theoretical justification of the topological signature from our previous work and explore new topological models for application purposes [1]. We will be considering the metrical representation of a boundary operator defined on the set of edges to the set of vertices in the context of an affine varieties so we can reconstruct the variety from an already defined algebraic topological space, let's illustrate by a first example, the following is a filtered simplicial table.

A quantification of the boundary operator obtained from Grobner and Buchberger algorithms using ideals as basis generators to solve a hidden polynomial equations system would be;

$$\begin{bmatrix} x_1^2x_2 & x_1^2x_2^2 & 0 & 0 & 0 & 0 & x_2^2x_1 & 0 & x_1^2x_2 \\ 0 & x_1^2x_2^2 & x_1^2x_2^2 & x_1^2x_2^2 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & x_1^2x_2^2 & x_1^2x_2^2 & x_1^2x_2^2 & 0 \\ 0 & 0 & 0 & x_1^2x_2 & x_1^2x_2 & 0 & 0 & x_1^2x_2 & x_1^2x_2 \\ 0 & 0 & 0 & 0 & x_1^2x_2^2 & x_1^2x_2^2 & 0 & 0 & 0 \\ x_1x_2^2 & 0 & x_1x_2^2 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

The two following theorems will play a central role in road mapping the inverse of the boundary and would also give us a justification to work in a commutative algebraic setting.

Theorem 1. (Strong Nullstellensatz) If K is an algebraically closed field and I is an ideal in $K[x_1, ..., x_n]$ then

$$I(V(I)) = \sqrt{I}$$

Theorem 2. (Ideal-Variety Correspondence) Let K be an arbitrary field; the maps

$$Affinevarieties \rightarrow ideals$$

and

$$ideals \rightarrow Affinevarieties$$

are inclusion reversing and

$$V(I(V)) = V$$

for all affine varieties V, if K is an algebraically closed then

$$Affinevarieties \rightarrow radicalideals$$

and

$$radicalideals \rightarrow Affinevarieties$$

are inclusion reversing bijections and inverses for each other.

Our free resolution is guaranteed from the following theorem.

Theorem 3. The boundary of a boundary vanishes, that is;

$$\partial_p \circ \partial_{p+1} = 0$$

Proof. We have;

$$\partial_{p-1}\sigma|_{[v_0,v_1,...,\hat{v_j},...,v_p]} = \sum_{i=0}^{j-1}(-1)^i \sigma|_{[v_0,v_1,...,\hat{v_i},...,\hat{v_j},...,v_p]} + \sum_{i=0}^{j-1}(-1)^{i-1}\sigma|_{[v_0,v_1,...,\hat{v_j},...,\hat{v_i},...,v_p]}$$

Correspondence to: Zakaria Lamine, Department of Mathematics, Ibnou Tofail University, Kenitra, Morocco, E-mail: zakarialamine2@gmail.com, zakaria.lamine1@uit.ac.ma

Received: 03-Sep-2024, Manuscript No. JPB-24-33809; **Editor assigned:** 05-Sep-2024, PreQC No. JPB-24-33809 (PQ); **Reviewed:** 19-Sep-2024, QC No. JPB-24-33809; **Revised:** 26-Sep-2024, Manuscript No. JPB-24-33809 (R); **Published:** 03-Oct-2024, DOI: 10.35248/0974-276X.24.17.673

Citation: Lamine Z, Mansouri MW, Mamouni MI (2024). A Mathematical Model for Protein Structure Prediction and Analysis. J Proteomics Bioinform. 17:673

Then

$$\partial_{p-1}\partial_p(\sigma)$$
$$= \partial_{p-1}\left(\sum_{j=0}^{p}(-1)^j\sigma\,|_{[v_0,v_1,\ldots,\hat{v_j},\ldots,v_p]}\right)$$
$$= \sum_{j=0}^{p}\sum_{i=0}^{j-1}(-1)^{i+j}\sigma\,|_{[v_0,v_1,\ldots,\hat{v_i},\ldots,\hat{v_j},\ldots,v_p]} + \sum_{j=0}^{p}\sum_{i=j+1}^{j-1}(-1)^{i+j-1}\sigma\,|_{[v_0,v_1,\ldots,\hat{v_j},\ldots,\hat{v_i},\ldots,v_p]}$$
$$= \sum_{i<j}(-1)^{i+j}\sigma\,|_{[v_0,v_1,\ldots,\hat{v_i},\ldots,\hat{v_j},\ldots,v_p]} + \sum_{i<j}(-1)^{i+j-1}\sigma\,|_{[v_0,v_1,\ldots,\hat{v_j},\ldots,\hat{v_i},\ldots,v_p]}$$
$$= 0$$

Let's now detail the computing part of the previous.

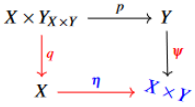
MATERIALS AND METHODS

Persistent diagram with different methods of construction

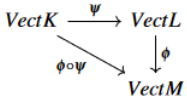
Let's consider the following revision from which we can derive a clear description of the class of linear statistical representations;

$$X \times Y$$

as universal components in a set theoretical context.

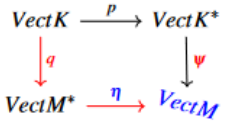


It is now sufficient to consider the pushout of the precedent diagram so the existence of our persistent diagram is guaranteed. Let us now involve more components to full-fill the definition, for that reason and to exploit efficiently theorems and proofs of the investigated theory, let us consider the functoriality of the main definition,



With ψ, φ are well defined vertex mappings between different set vertices contained in a filtered simplicial complexes, we should also mention that no theoretical frame or applied one is given in the literature for a comparison between kernel density estimation construction *vs* alpha complex one of the persistent diagram in topological data analysis.

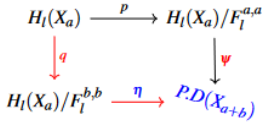
To be able to visualize the filtration process, one needs to consider the pullback given by,



Then given a sequence of inclusions of topological spaces

$$X_a \subseteq X_b \subseteq \ldots \subseteq X_{a+b}$$

and its homology groups cautioned by their tames, a persistent diagram up to isomorphism is given by the following;



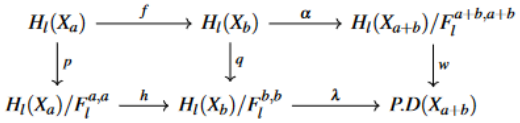
The inclusions of topological spaces induces immediately an inclusion between the cautioned spaces, we now can be sure from the greatest lower bound which is

$$H_l(X_a)/F_l^{a,a} \times H_l(X_a)/F_l^{b,b}$$

This gives *a* theoretical frame to construct our confidence sets intervals. We should mention before getting in the proposed probabilistic models or the way they are writing that computer simulations nowadays made the theoretical frame quite flexible but less deliberate, especially when a new theory is proposed. This is the case with persistent diagrams. We should also mention that persistent diagrams are either derived from a learning process or functional summaries within a larger Hilbert space, for that reason one should investigate how the replicated persistent diagrams can be generated and what makes it different from other traditional constructions, principal component analysis as an example. To prove existence and definition of a replicated persistent diagram, we will be solving the problem of replication by investigating the behaviour of a persistent diagram near its greatest lower bound given by following [2].

$$H_l(X_a)/F_l^{a,a} \times H_l(X_a)/F_l^{b,b}$$

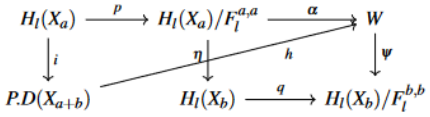
From the already defined inclusion of topological spaces, we derive the following commutative diagram;



then we induce by using relative homology the following exact sequence;

$$Kerp \xrightarrow{f} Kerq \xrightarrow{g} Kerw \xrightarrow{u} cokerp \xrightarrow{f'} cokerq \xrightarrow{g'} cokerw$$

Which means the caution could be defined for the whole inclusion, then a replicated persistent diagram is theoretically guaranteed. To fulfill the definition we consider the following diagram.



The uniqueness of our persistent diagram to conclude the definition depends on a factorization of the previous in the functor *h* back to the previous relative sequence we have.

$p, q \in H^*$ are well defined projections which implies $H \in W$ it is now sufficient to prove $Imp \in W$ or $Imp \in H_l$ or $(X_a)/F_l^{b,b} \in H^*$ the second inclusion is given by construction or in $H_l(Xa)/F_l^{b,b}$ every map calculate a homology within $F_l^{b,b}$ we confirm that W has the same topological degree as $P.D(X_{a+b})$ which gives the commutativity of the diagram. We conclude the uniqueness of $P.D(X_{a+b})$ then P.D(X) for any topological space (X) with some degree p.

Being said, the immediate way to start is building a confidence set interval for $W_\infty(\hat{P}, P)$

With $\hat{P}$ is an estimate of the persistent diagram constructed from a sample.

W_∞ is the bottleneck distance, we consider for that reason the theorem.

Theorem 4. Let f, g : K → R be monotone functions. Then;

$$W_p\left(Dgm_k(f), Dgm_k(g)\right) \leq |f-g|_p$$

for a homology dimension k we have;

$$W_p\left(Dgm_k(f), Dgm_k(g)\right)^p \leq \sum_{dim(\sigma)\in k,k+1} |f(\sigma)-g(\sigma)|^p$$

We then bound

$$H(S, M)$$

such that H is the Hausdorff distance;

$$H(K, M) = inf\{\varepsilon : K \subset M \oplus \varepsilon and M \subset K \oplus \varepsilon\}$$

to obtain a bound on

$$W_\infty(\hat{P}, P)$$

with $\varepsilon = Z(VectM^*)$

We can now easily define a $1-\alpha$ confidence set interval for the bottleneck distance;

$$W_\infty(\hat{P}, P)$$

that is;

$$\lim_{n\to\infty} infP(W_\infty(\hat{P}, P) \in [0, p_n]) \geq 1-\alpha$$

With p_n an adequate statistical descriptor of $\hat{P}$ the last step is to find α such that

$$\lim_{n\to\infty} supP(H(S_n, M) > c_n) \leq \alpha$$

Then the set of persistent diagrams are given.

$(\varepsilon \oplus C_n)$ such that; C_n is the confidence set related to $\hat{P}$.

Being said, we get a confirmed theoretical frame to start the statistical study that involve point clouds representing atoms lying in a high dimensional space with a hidden locally Euclidean manifold. The next step consists of presenting algorithms derived from the previous result mentioned in the introduction, which is persistent homology of filtered complex is nothing but the regular homology of a graded module over a polynomial ring [3-6].

Polynomial solutions of boundary operators

Boundary and cycles modules: The concept of boundary and cycles is theoretically formalized in the previous definition of a persistence homology, homology gives a description of the set of cycles, by using the caution over the set of boundaries, which also means by persistence, preserving the cycles that are not boundaries.

$$H_k^{l,p} = Z_k^l / (B_k^{l+p} \cap Z_k^l)$$

In our context, cycles are the significant topological signatures of all types including loops and loops of loops, holes and cavities and so on. Let's now compute our homologies, as already mentioned in the introduction persistent homology of filtered complex is nothing but the regular homology of a graded module over a polynomial ring, our module is defined over the n graded polynomial ring;

$$A^n = k[x_1, ..., x_n]$$

with standard grading

$$A_v^n = k.x^v, v \in N^n$$

then

$$R = A^n$$

Then our vector of polynomials is writing as $[a_1, ..., a_m]^T$, a_i is a polynomial where the matrix M_{i+1} for ∂_{i+1} has m_i rows and m_{i+1} columns where mj stands for the number of $j-simplices$ in the complex, a_i is the i^{th} column in M_{i+1} thus we can separate polynomials from the derived coefficients, let

$$A = (a_1, ..., a_{m_{i+1}}), a_i \in R^{m_i}$$

Where a_i is the i^{th} column in M_{i+1} one now can write a polynomial vector a in a submodule in term of some basis A as in

$$< A >= \sum_{j=1}^{m_{i+1}} q_j a_j / q_j \in R$$

to get a final result computing ∂_{i+1}. Things seems easier for the cycle submodule, which is a submodule of the polynomial module. As previously, this time ∂_i has m_{i-1} rows and m_i columns,

$$A = (a_1, ..., a_{m_i}), a_i \in R^{m_{i-1}}$$

Where a_i is the i^{th} column in the matrix, the set of all $[q_1, ..., q_{mi}]^T$ such that

$\sum_{i=1}^{m_i} q_i a_i = 0$ is a R submodule of R^{m_i} which is the first Syzygy module of $(a_1, ..., a_{mi})$. A set of generators of the previous would finish the task, then finally to compute our homologies it suffices to verify whether the generators of the Syzygy submodule are in the boundary submodule.

Solving the problem of the boundary within a variety would consists of solving all edges and vertices within a set of polynomials equations without losing topological significance. The inverse inclusion would give an exact sequence for the boundary operators. The problem then takes the form of a free resolution, so we have the following computation.

Computation of homologies and rank invariant: Let's consider the polynomial module R^m with the standard basis $e_1, ..., e_m$ where e_i is the standard basis vector with constant polynomial 0 in all positions except 1 in position i, *min* R^m is of the form $x^u e_i$ for some i and we say m contains e_i. For, $u, v \in N^n$ $u > v$, *if* $u - v \in Z^n$ the left most nonzero entry is positive, this gives a total order on N^n as an example (1, 4, 0)>(1, 3, 1) since (1, 4, 0)−(1, 3, 1)=(0, 1, 0) the left most nonzero is 1, for two monomials x^u, x^v in R, $x^u > x^v$ if $u > v$ which gives a monomial order on R we then extend the order on R^m by using $x^u e_i > x^v e_j$ if $i<j$ or if $i=j$ and $x^u > x^v$, $r \in R^m$ can be written in a unique way, as a k linear combination of monomials m_i;

$$\sum_i^{m_i} c_i m_i$$

Where $c_i \in K$, $c_i \neq 0$ and m_i ordered according to monomial order. As an example, if we consider

$$f = k[7x_1x_2^2, 3x_1 - 5x_3^3]^T \in R^2$$

Then we can write f in terms of the standard basis

$$f = 7[x_1x_2^2, 0]^T - 5[0, x_3^3]^T + 3[0, x_1]^T = 7x_1x_2^2e_1 - 5x_3^3e_2 + 3x_1e_2$$

We then extend operations such as least common multiple to monomials in R and R^m we summarize them by saying

$$m / n = x^u / x^v = x^{u-v}$$

After a division, we get

$$a = \sum_1^t q_i a_i + r$$

So, if $r=0$ then $a \in <A>$ so the division is not a sufficient condition, for that reason we use Grobner basis then by forcing the leading terms to be equal we get a sufficient condition. For unicity and minimality, we reduce each polynomial in G by replacing $g \in G$ by the remainder of $g/(G-g)$ then $im\partial_{i+1}$ is well computed.

Still to compute generators for the Syzygy submodule, we compute a Grobner basis.

$$A = \{a_1, ..., a_s\}$$

For <A> where the ordering is the monomial one, we then follow the same process as for im∂_{i+1} we get

$$S(a_i,a_j)=\sum_1^s q_{ijk}g_k$$

with g_k elements of the Grobner we need now a Grobner basis for

$$SYZ(a_1,...,a_s)$$

which can be obtained by using Schreyer's theorem, guaranteeing the existence of

$$S_{ij}=\frac{h_{ij}}{LT(a_i)}\varepsilon_i-\frac{h_{ij}}{LT(a_j)}\varepsilon_j-q_{ij}\in R^S$$

otherwise, we use this basis to find generators

$$SYZ(g_1,...,g_s)$$

for a metrical representation we consider elements a_i and g_i from S as columns of a given M_A and M_G respectively, the two basis generate the same module. ∃A, *B* such that $M_G=M_AA$, $M_A=M_GB$ with each column of M_A is divided by M_G since M_G a Grobner basis for M_A. We conclude, there is a column in *B* for each column $a_i \in M_A$ that can be obtained by division of a_i by M_G. Let;

$$S_1,\ ...,\ S_t$$

be the columns of the $t \times t$ matrix $I_t - AB$. Then;

$$SYZ(a_1,...,a_t)=<AS_{ij},S_1,...,S_t>$$

Then the *Ker*∂_i is computed. Finally we need to compute the caution H_i given im∂_{i+1}=<G> and *Ker*∂_i=*SYZ*(a_1, ..., a_t). We divide every column in *Ker*∂_i by *im*∂_{i+1} using the same process as in computing *im*∂_{i+1} if the remainder is non zero we add it both to im∂_{i+1} and H_i. Therefore, we count only unique cycles. We obtain for the previous bifiltration the following homogenous matrix for ∂_1 So M_{11} is obtained by cautioning

$$j:x_1^2x_2^2 \text{ by } A:x_1^2x_2 \text{ we get } M_{11}=x_2$$

and so on, the full matrix then has the form

$$\begin{bmatrix} x_2 & 1 & 0 & 0 & 0 & 1 & 0 & x_1^2x_2 \\ 0 & x_2 & x_1 & x_2 & 0 & x_2 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & x_1x_2^2 & 0 \\ 0 & 0 & 0 & 1 & x_1 & 0 & x_1x_2 & x_1^2x_2 \\ 0 & 0 & 0 & 0 & x_1x_2 & 0 & 0 & 0 \\ x_1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

To compute the rank invariant, we can use the multigraded approach, then if we take the previous bifiltration, matrices for SYZ(G_1) and Grobner of Z_1 for ∂_1 are obtained as previously,

Multi-filtered dataset: In topological data analysis, a multifiltered data set can be defined as;

Definition 1. (*S*, {f_j} *j*), where *S* is a finite set of d-dimensional points with *n – 1* real-valued functions.

$$f_j:S\to R$$

Defined on it, for *n>1*. We assume our data is a multifiltered dataset (*S*, {f_j} *j*).

In the following definitions, the calculations are made in commutative algebraic setting, this induces an order on the multifiltration, which can be viewed as an action of a ring over a module plus an inclusion maps relating copies of vertices within complexes, we will be using the ring of polynomials to relate the chain groups in the different grades of the module as the following;

$$0\xrightarrow{i}C_p(K)\xrightarrow{\partial_p}C_{p-1}(K)\xrightarrow{\partial_{p-1}}...\xrightarrow{i}C_0(K)\xrightarrow{\partial_0}0$$

with

$$C_i=\oplus_u C_i(K_u)$$

For that purpose, let us detail the definition.

Definition 2. A p-dimensional simplex or $p-simplex\ \sigma^p=[e_0, e_1, ..., e_p]$ is the smallest convex set in a Euclidean space R^m containing the *p+1* points e_0, ..., e_p;

$$\Delta^P=\{(t_0,...,t_p)\in R^{p+1}:\sum_{i=0}^p t_i=1 and\ ti\geq 0\ for\ all\ i\ =\ 0,\ ...,\ p\}$$

Another interesting and explicit description of persistent homology *via* visualization of barcodes can be found in [7]. We suggest here a concise precise definition *via* classification theorem.

Remark 1 (Persistence modules). We apply the "homology functor" to the filtered chain complexes, so we get our "homology groups" category [8,9]. This can be viewed as;

$$0\xrightarrow{i}H_p(K)\xrightarrow{\partial_p}H_{p-1}(K)\xrightarrow{\partial_{p-1}}...\xrightarrow{\partial_1}H_0(K)\xrightarrow{\partial_0}0$$

Where → denotes the inclusion map.

For a finite persistence module C with filed F coefficients

$$H_*(C;F)\cong\oplus_i x^{t_i}.F[x]\oplus(\oplus_j x^{r_j}.(F[x]/(x^{S_j}.F[x]))$$

that are the quantification of the filtration parameter over a field, with clear description [10].

Definition 3. The p-persistence *k-th* homology group

$$H_k^{l,p}=Z_k^l/(B_k^{l+p}\cap Z_k^l)$$

well defined since B_k^{l+p} and Z_k^l are subgroups of C_k^{l+p}.

Let us consider the previous bi filtration from the introduction; we assume the computation are in

$$Z\oplus Z$$

and u_1=(0, 2), u_2=(0, 1), u_3=(0, 0), u_4=(1, 2), u_5=(1, 1), $u_{6=}$(1, 0), u_7=(2, 2), u_8=(2, 1), u_9=(2, 0), u_{10}=(3, 2), u_{11}=(3, 1), u_{12}=(3, 0) to be read from top to the bottom.

In this example, we have F_4 in grade (0, 0),

$F_5=x_1 \times F_4$ in grade (0, 0).

$F_6=x_2 \times F_5= x_1 \times x_2 \times F_4$ in grade (1, 1) and so on, then ∂_1 as from

$$0\xrightarrow{i}C_p(K)\xrightarrow{\partial_p}C_{p-1}(K)\xrightarrow{\partial_{p-1}}...\xrightarrow{\partial_1}C_0(K)\xrightarrow{\partial_0}0$$

can be computed

$$\begin{bmatrix} x_1^2x_2 & x_1^2x_2^2 & 0 & 0 & 0 & 0 & x_2^2x_1 & 0 & x_1^2x_2 \\ 0 & x_1^2x_2^2 & x_1^2x_2^2 & x_1^2x_2^2 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & x_1^2x_2^2 & x_1^2x_2^2 & x_1^2x_2^2 & 0 \\ 0 & 0 & 0 & x_1^2x_2 & x_1^2x_2 & 0 & 0 & x_1^2x_2 & x_1^2x_2 \\ 0 & 0 & 0 & 0 & x_1^2x_2^2 & x_1^2x_2^2 & 0 & 0 & 0 \\ x_1x_2^2 & 0 & x_1x_2^2 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

Computation of homologies and rank invariant for atoms point cloud: Being said gives a clear road map to start hypothesizing over a real data set. For that reason, let us consider a folding protein that constitutes *N* particles and has the spatiotemporal complexity of $R^{3N} * R^+$. We assume that our system can be described as a set of *N* nonlinear oscillators of dimension $R^{nN} * R^+$, where n is the dimensionality of a single nonlinear oscillator. We will be using data from the freely data bank of Protein Data Bank (PDBs), the molecule in consideration has 1cos as an ID. Our point cloud lying in a $R^{3.700}$, coordinates of atoms are considered as the input of our multidimensional filtration (Figure 1).

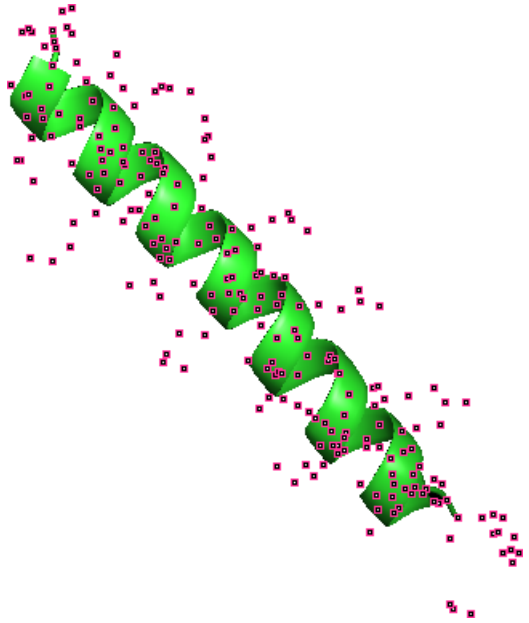


Figure 1: The all-atom representation of an alpha helix.

We obtain in a first sight the following topological signatures which means our final result is a three dimensional simplex (Figure 2).

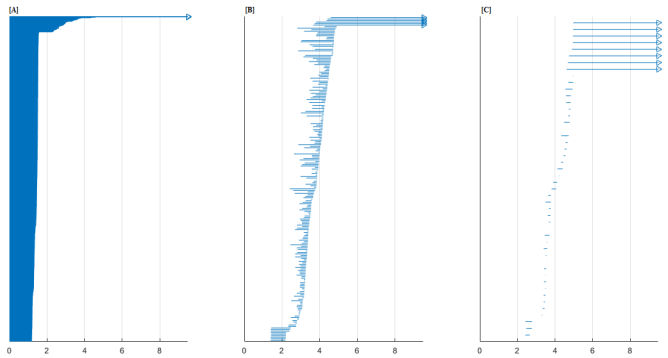


Figure 2: Topological fingerprints of the molecule. **Note:** A) cois5 (dimension 0); B) cois5 (dimension 1); C) cois5 (dimension 2).

To simplify the task let us consider the alpha carbon atoms of our molecule (Figure 3).

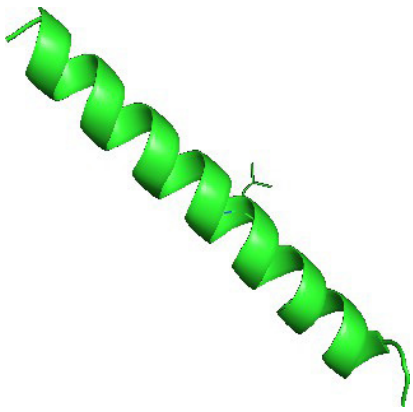


Figure 3: Coarse-Grained (CG) representation of an alpha helix generated from a protein of Protein Data Bank (PDB) ID 1cos.

The topological signature was given in Figure 4.

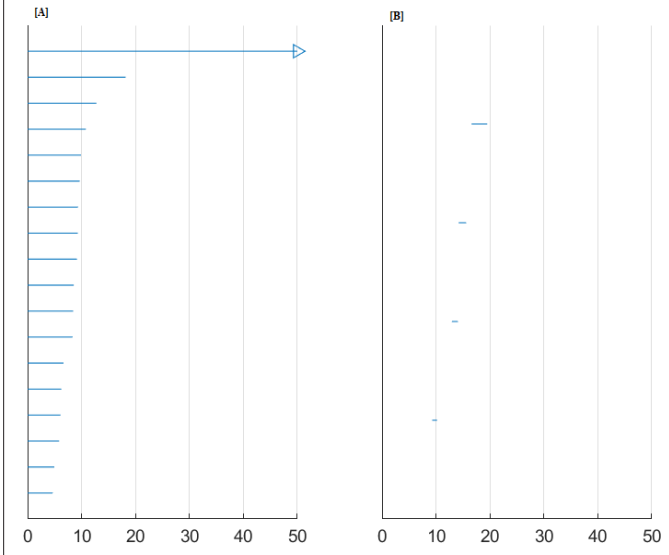


Figure 4: Fingerprint of Coarse-Grained (CG) representation of an alpha helix generated from a protein of Protein Data Bank (PDB) ID 1cos. **Note:** A) 1cos (dimension 0); B) 1cos (dimension 1).

This means our final result when the end of the multifiltration is a one dimensional simplex, with eighteen vertex at the beginning of the multifiltration; u_1=(3, 20, 21), u_2=(3, 19, 21), u_3=(4, 21, 22), u_4=(3, 21, 23), u_5=(3, 19, 23), u_6=(3, 20, 24), u_7=(4, 21, 23), u_8=(3, 22, 25), u_9=(4, 20, 22), u_{10}=(3, 21, 24), u_{11}=(3, 22, 26), u_{12}=(3, 23, 26), u_{13}=(4, 25, 26), u_{14}=(4, 24, 25), u_{15}=(4, 19, 25), u_{16}=(3, 23, 19), u_{17}=(4, 23, 26), u_{18}=(4, 22, 27).

We get after calculations the following matrix;

$$\begin{bmatrix} 1 & 1 & 0 & 0 & 0 & 1 & 0 & 1 \\ 0 & 1 & 1 & 1 & 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 1 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 1 & 1 & 0 & 1 & 1 \\ 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 \\ 1 & 0 & 1 & 0 & 0 & 0 & 1 & 1 \end{bmatrix}$$

which means the final shape conserve only one type of homology, with four loops as generators. Let us now involve more parameters, we consider decreasing radial basis functions. The general form is;

$$c_{ij} = \omega_{ij}\Phi(r_{ij},\eta_{ij})$$

Where, ω_{ij} is associated with atomic types, then a generalized exponential kernel has the form;

$$\Phi(r,\eta) = e^{-(r/\eta)k}$$

k>0 one then can construct the following matrix.

$$M_{ij} = \{_0^{1-\Phi(r_{ij},\eta_{ij})}\ If_{i=j}^{i\neq j}\} \text{ with } \Phi(r,\eta) = \frac{1}{1+(r/\eta)^v}$$

This matrix can easily be obtained following the division algorithm mentioned in the previous section. By considering, xyz coordinates of atoms as the input of the multifiltration, and then the result can be used as the input for the persistent homology calculations following the same process. This clearly shows the path for an easiest extraction of a shape of a protein, since the traditional methods use many complicated parameters to build matrices supposed to rebuild the geometric conformation as the case of molecular nonlinear dynamics and flexibility rigidity index involving exponential kernels with parameters. For more

enlightenment through an interesting detailed investigation of topology function relationship paradigm of proteins [3,11].

RESULTS

As we have already mentioned in the previous section a full description of persistent homology can be obtained following; persistent homology of filtered complex is nothing but the regular homology of a graded module over a polynomial ring. The computation is also easy following; a division algorithm then a Buchberger algorithm to seek generators then basis (ideals) for modules. The final step for a statistical analysis is a quantification of the result of the second section to figure out the so-called replicated persistent diagrams. We can observe the significant topological difference between tertiary structures and secondary structures from Figure 5; the interesting task would be a separation between the alpha helices and beta sheets. Those can be literally expressed as the length of the coefficients lying in the metrical representation of our computed quotient Hi [2,12].

The total loss function incorporating homology into the learning process is given by;

$$L(\theta) = \sum_{i=1}^{n} Loss(f(\mathbf{x}_i;\theta), y_i) + \lambda \sum_{j=1}^{m} h_j$$

Where

- L (θ) is the total loss function of the model.
- Loss (f (x_i; θ), y_i) is the standard loss function for the i^{th} data point.
- f (x_i; θ) is the model's prediction for input xi with parameters θ .
- y_i is the true label for the i^{th} data point.
- h_j is the homology coefficient for the j^{th} feature or level.
- λ is the regularization parameter that controls the weight of the homology term.

After running the model through our dataset, we get a folding process describing the behavior of different types of homologies through variation of our Gaussian probability distribution.

For a neural network with a single hidden layer, the learning function can be summarized as follows;

$$\mathbf{y}_{pred} = \sigma(\mathbf{W}_3\sigma(\mathbf{W}_2\sigma(\mathbf{W}_1\mathbf{x}+\mathbf{b}_1)+\mathbf{b}_2)+\mathbf{b}_3)$$

Where,

- $\sigma(z)$ is the activation function (e.g., sigmoid for binary classification, soft-max for multi-class classification).
- $L(y, y_{pred})$ is the loss function (e.g., binary cross-entropy or categorical cross-entropy).

The parameter updates using gradient descent are given by;

$$\mathbf{W}_i \leftarrow \mathbf{W}_i - \eta \frac{\partial L(\mathbf{y}, \mathbf{y}_{pred})}{\partial \mathbf{W}_i}$$

$$b_i \leftarrow b_i - \eta \frac{\partial L(\mathbf{y}, \mathbf{y}_{pred})}{\partial b_i}$$

Where,

- η is the learning rate.
- $\frac{\partial L}{\partial \mathbf{W}_i}$ and $\frac{\partial L}{\partial b_i}$ are the gradients of the loss with respect to weights and biases.

Statistical summary of models is given (Table 1, Figures 5-7).

Table 1: Comparison of DeepCNF and DNN-Pred models.

Metric	DeepCNF (2018)	DNN-Pred (2019)
Accuracy	~ 80%	75%-78%
Precision	High for residue-level prediction	Good for domain classification
Recall	High for secondary structure tasks	Good for protein interactions
Complexity	High (CNN+CRF)	Moderate (MLP)
Training data	Large PDB datasets	CASP and other benchmarks
Computational cost	High due to CRFs and CNNs	Moderate due to MLP

Note: DeepCNF: Deep Convolutional Neural Fields; DNN-Pred: Deep Neural Network-based Prediction; CNN: Convolutional Neural Network; CRF: Conditional Random Field; MLP: Multilayer Perceptron; PDB: Protein Data Bank; CASP: Critical Assessment of Structure Prediction.

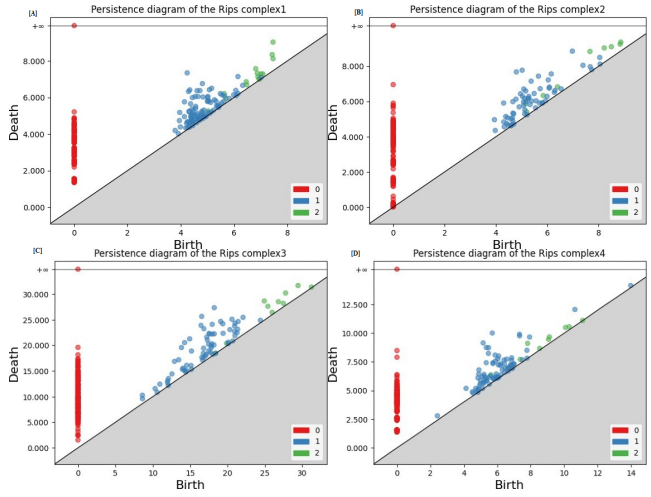


Figure 5: Persistent diagrams generated from xyz distributions of the alpha carbon atoms. **Note:** A) Protein Data Bank (PDB) id 1cos; B) PDB id 2jox; C) PDB id 6idd and D) PDB id 1dgv.

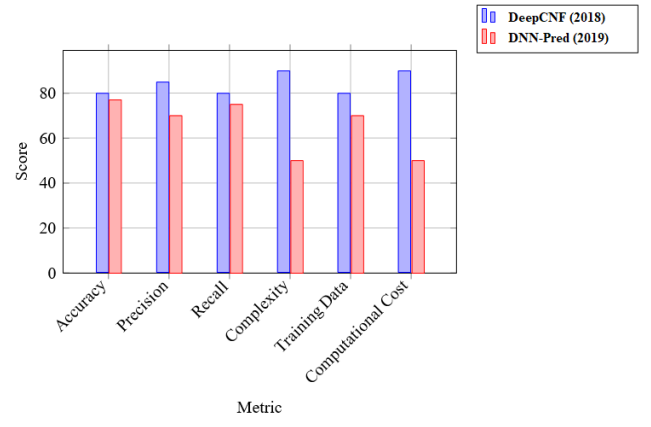


Figure 6: Comparison of Deep Convolutional Neural Fields (DeepCNF) and Deep Neural Network-based Prediction (DNN-Pred) models.

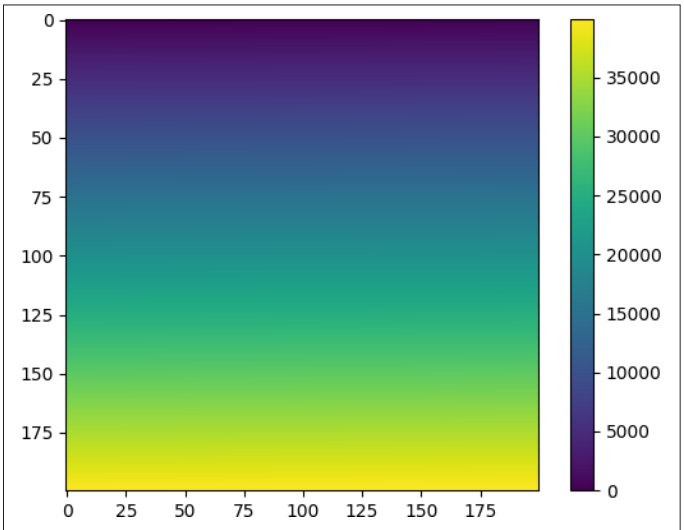


Figure 7: Behaviour of homologies through variation of the distribution.

Experimental procedures

In the all-atom model, atoms are considered the same; each atom is associated with the same radius in the distance-based filtration. The stream will be constructed for the point cloud data, which is the xyz, coordinates of the all atom representation (Figure 8). The size is not too large to choose a landmark selector, so we will simply build a Vietoris-Rips stream. We can choose a better filtration but for the limited computation power, we stick with the value of 8. In this case a Vietoris-Rips complex is largely sufficient to decipher the topological fingerprints (a small data set) so there is no need to use a landmark selector, which can be seen in the code shown below [13-16].

Figure 8: Example of structure used in our training data set with Protein Data Bank (PDB) Id 6IDD.

```
>> size (ecos)
ans =
696       3
>> max_dimension = 3;
>> max_filtration_value = 8;
 >> num_divisions = 1000;
>> stream =  api.Plex4.create
Vietoris-Rips Stream (ecos, max_dimension , . . . max_filtration_v
alue, num_divisions);
>> num_simplices=stream.getSize()
numsimplices=3259289
>> persistence=api.Plex4.get
ModularSimplicialAlgorithm (max_dimension, 2);
>> options.filename = 'cois';
>> options.maxfiltrationvalue
= max filtration value;
>> option max_dimension=max_dimension-1;
>> options.side_by_side=true;
>> plot_barcodes(intervals,options);
```

We utilize the Coarse-Grained (CG) with each amino acid represented by its Alpha Carbon (Cα) atom. The simplices are constructed which is helpful for the detection of the helix structure, so the corresponding barcode is simplified. As the last construction a Vietoris-Rips stream is largely sufficient to decipher the topological features of our data which is an 18 points in a 3-dimensional space. A part of the Matlab© code is shown below.

```
>> load  ecos 1
>> size (ecos)
ans =
18        3
>> max_dimension=2;
>> max_filtration_value=2 3;
>> num_divisions=1000;
>> stream=api.Plex4.createVietorisRipsStr
eam (ecos, max_dimension, . . . max_filtration_v
alue, numdivisions);
>> options.filename = 'coiis2';
>> options.maxfiltrationvalue=maxfiltrat
ionvalue;
>> options.max dimension=max dimension-1;
>> persistence = api.Plex4.get
ModularSimplicialAlgorithm (max_dimension, 2);
>> options.sidebyside = true;
>> intervals =persistence.computeInterva
ls (stream);
>> plotbarcodes(intervals, options);
```

for the beta sheet construction, we use the following:

```
>> loadfinbeta
>> max_dimension=2;
>> max_filtration_value=5;
>> num_divisions=1000
 numdivisions=
1000
>> stream=api.Plex4.createVietorisRipsStre
am (beti001, max_dimension, . . . max_filtration_v
alue, num_divisions);
>> numsimplices=stream.getSize() numsim
plices=
149776
```

```
>> persistence = api.Plex4.get
ModularSimplicialAlgorithm(max_dimension, 2);
>> intervals = persistence.computeIntervals(stream);
>> options.filename = '1bet';
>> options.maxfiltrationvalue=max_filtration_value;
>> options.max_dimension=max_dimension-1;
>> options.sidebyside = true;
>> plotbarcodes(intervals,options);
```

Then we simplify without loosing topological significance by using the following:

```
>> loadbety
>> size(beti01) ans=
 24      3
>>max_dimension=2;
>> max_filtration_value =20;
>> numdivisions=1000 numdivisions=
1000
>> stream=api.Plex4.
createVietorisRipsStream
(beti01, max_dimension, ...
max_filtration_value, num_divisions);
>> numsimplices=stream.getSize() numsim
plices=
410
>> persistence = api.Plex4.get
ModularSimplicialAlgorithm(max_dimension, 2);
>> intervals=persistence.computeIntervals(stream);
>> options.filename = 'bet';
>> options.maxfiltrationvalue=
max_filtration_value;
>>options.max_dimension=max_dimension-1;
>> options.side_by_side=true;
>> plotbarcodes (intervals, options);
```

```
# -*- coding: utf-8 -*-
""" Untitled1.ipynb
 """
import numpy as  np
import matplotlib.pyplot as plt
import numpy as np
impo
rt seaborn as sns
from matplotlib import colors
from mpltoolkits.mplot3d import Axes3D
import matplotlib.pyplot as plt
from pylab import*
from array import array
pip install pdbreader
import pdbreader
pdb=pdbreader.readpdb("/content/2jox.pdb")
from google.colab import drive
drive.mount('/content/drive')
"""#  Nouvelle section """
for key in pdb:
print (key)
ATOM = pdb ['ATOM']
type (ATOM)
for key in ATOM:
print (key)
matrix = [ATOM ['x']],
ATOM ['y'],
ATOM ['z']]
print (matrix)
table=np.empty ((696, 3))
print (table)
for i in range (0,696):
table [i] = [np.array (row [i]) for row in matrix]
print (table [i])
a=np.asarray (table)
np.savetxt ("/content/matrixNOTordered.csv", a, delimiter=",")
tabEX = table
print (tabEX)
a=np.asarray(tabEX)
np.savetxt("/content/cc.csv", a,delimiter=" , ")
def arrange(tabEX, n, m):
for i in range (0, m):
line = tabEX [i] - tabEX [i+1]
for j in range (0, n):
if line [j]<0:
temp = tabEX [i]
tabEX [i] = tabEX [i+1]
tabEX [i+1] = temp
break
elif line [j]>0:
break
elif i<18: print (tabEX [i])
else: continue
return tabEX
print (tabEX)
a=np.asarray(tabEX)
np.savetxt("/content/kk.csv", a,delimiter=" , ")
from typing import KeysView
table au = np.empty ((696,3))
for i in range (696, 3):
for j in range (696, 3):
```

```
for k in range (0, 2):
t a b l e a u [ i ] = tabEX [j] [k] * tabEX [j] [k+1]
p r i n t (t a b l e a u)
from typing import KeysView
Grobner = np.empty ((696,3))
def  Grobnera      (tableau, X, Y, Z):
for j in range (696, 3):
for k in range (0, 2):
Grobner [j] [k] =   tableau [j] [k]/table a u [j] [k+1]
return Grobner
"""#  Nouvelle section"""
a=np.asarray (Grobner)
np.savetxt ("/content/kk.csv", a, delimiter=",")
tableau.shape
def Gaussian_kernel_matrix (Grobner, sigma):
    distances = np.sum ((Grobner [:,np.newaxis] − X) ** 2, axis = −1)
    kernel_matrix = np.exp (−distances/(2*sigma**2))
return kernel_matrix
X = Grobner
sigma =1
kernel_matrix = gaussian_kernel_matrix (X, sigma)
print (kernel_matrix)
a=np.asarray (kernel matrix)
np.savetxt ("/content /matrixNOTordered.csv", a, delimiter=",")
type (Grobner)
import numpy as np
import matplotlib.pyplot as plt
import numpy as np
def heatmap2d (arr: Grobner):
p l t . i m s h o w (a r r , c m a p = ' v i r i d i s ')
p l t . c o l o r b a r ()
p l t . s h o w ()
t e s t a r r a y = n p . a r a n g e (2 0 0 * 2 0 0) . r e s h a p e (2 0 0 , 2 0 0)
heatmap2d (test_array)
```

Linking different types of homologies to figure out the full simulation of persistent homology will be done using a training parameter under a reinforcement learning approach. We will be building neural networks using numpy library from python; with only two hidden layers our model seems to hide greater strategies in capturing alpha shapes from a twenty data set of matrices representing *Hi* as our training data set (Figure 9); to figure out probabilities we use the following softmax activation function [10].

$$\begin{bmatrix} 6.91834797e-310 \\ 3.72965666e+242 \\ \vdots \\ 1.44798216e-023 \end{bmatrix} \rightarrow \frac{e^{z_i}}{\sum_{i=1}^{k} e^{z_j}} \rightarrow \begin{bmatrix} 0.9 \\ 0.2 \\ \vdots \\ 0.1 \end{bmatrix}$$

```
[[ 6.91834797e-310  6.91834797e-310  4.81257868e-310]
 [ 4.81257868e-310  5.62740771e-321  0.00000000e+000]
 [ 3.34481499e-298  4.81257817e-310  1.86780318e-300]
 ...
 [ 3.72965666e+242  6.91834664e-310  6.91834664e-310]
 [-3.59898896e+288  6.91834664e-310  6.91834664e-310]
 [ 1.44798216e-023  6.91834664e-310  6.91834664e-310]]
```

Figure 9: Each row is a sample in our data set.

```
pip install tensorflow biopython numpy pandas
import numpy as np
import pandas as pd
from Bio import PDB
def load_pdb_data (file_path):
parser = PDB.PDBParser()
structure = parser.get_structure ('protein', file path)
atom_data = []
for model in structure:
for chain in model:
for residue in chain:
for atom in residue:
atom_data.append ([atom.get name (), atom.coord [0],
atomdf = pd.DataFrame (atom data, columns =['AtomName', 'X',  'Y',
return atom_df
#Example usage
pdb_data = load_pdb_data ('example.pdb')
print (pdb data.head ())
# Data preparation
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
def prepare_data (df, labels):
features = df [['X', 'Y','Z ']]. v a l u e s
s c a l e r = S t a n d a r d S c a l e r ()
f e a t u r e s = s c a l e r . f i t t r a n s f o r m (f e a t u r e s)
X _ t r a i n , X _ t e s t , y _ t r a i n , y _ t e s t = t r a i n _ t e s t _ s p l i t (f e a t u r e s , l a b
return X _ t r a i n,  X _ t e s t , y _ t r a i n , y _ t e s t
#Example labels (you would need actual labels for your dataset)
labels=np.random.randint (0, 2, len (pdb data)) # Example: Binary cl
X_train, X_test,  y_train, y_test=prepare_data (pdb_data, labels)
#Define and train the model
import tensorflow as tf
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense, Dropout
def build_model (input_shape):
model=Sequential ([
Dense (64, activation='relu', input_shape=(input_shape,)),
Dropout (0.5),
Dense (32, activation='relu'),
Dense (1, activation='sigmoid')
```

```
])
model.compile (optimizer='adam',
loss='binary_crossentropy',
metrics=['accuracy'])
return model
#Example usage
model=build model (X_train.shape [1])
model.summary ()
#Train the model
history=model.fit (X_train, y_train, epochs=10, batch_size=32, valid)
#Evaluate the model
loss, accuracy=model.evaluate (X_test, y_test)
print (f " Test Loss: {loss}")
print (f" Test Accuracy: {accuracy}")
```

DISCUSSION

It was out of the scope of this proposition to deal theoretically with the use of statistical tests on the set of barcodes, but the application shows clearly that the method can surpass a simple statistical approach and instead of conducting a molecular dynamic simulation it is easier to use existing information from models to construct a quantified sequence of barcodes then to look for its convergence limit. We can find interesting productions in the literature but none exploited fully persistent homology far from being a statistical tool. An interesting attempt by using dynamical distances was made by Peter Bubenik and collaborators, but couldn't theoretically justify barcodes as a statistical observation, instead it gives birth to a new functional tool which is persistent landscapes [17].

CONCLUSION

This work is providing a complete roadmap for persistent homology and application to protein structure design, prediction and analysis. Persistent homology is a powerful tool in the field of computational biology, allowing researchers to analyze complex biological structures with greater accuracy and efficiency. In order to fully understand the fundamental concept, the mathematical model is thoroughly explained. To get familiarized with the axiomatic idea, the full mathematical model is detailed, as already mentioned in the computational part and give a stochastically approval to our work but without getting in the details of the calculations and the theoretical justifications.

REFERENCES

1. Lamine Z, Mamouni MI, Mansouri MW. A topological data analysis of the protein structure. Int J Anal Appl. 2023;21:136.
2. Xia K, Opron K, Wei GW. Multiscale multiphysics and multidomain models—Flexibility and rigidity. J Chem Phys. 2013;139(19):194109.
3. Buchet M, Chazal F, Oudot SY, Sheehy DR. Efficient and robust persistent homology for measures. Comput Geom. 2016;58:70-96.
4. Edelsbrunner H, Morozovy D. Persistent homology: Theory and practice. European Congress of Mathematics. 2014:31-50.
5. Ichinomiya T, Obayashi I, Hiraoka Y. Protein-folding analysis using features obtained by persistent homology. Biophys J. 2020;118(12):2926-2937.
6. Kovacev-Nikolic V, Bubenik P, Nikolić D, Heo G. Using persistent homology and dynamical distances to analyze protein binding. Stat Appl Genet Mol Biol. 2016;15(1):19-38.
7. Carlsson G. Topology and data. Bull Amer Math Soc. 2009;46(2):255-308.
8. Liu J, Xia KL, Wu J, Yau SS, Wei GW. Biomolecular topology: Modelling and analysis. Acta Math Sin Engl Ser. 2022;38(10):1901-1938.
9. Hatcher A. Algebraic Topology. 2005.
10. Zomorodian A, Carlsson G. Computing Persistent Homology. Proceedings of the twentieth annual symposium on Computational geometry. 2004. 347-356.
11. Xia K, Wei GW. Stochastic model for protein flexibility analysis. Phys Rev E Stat Nonlin Soft Matter Phys. 2013;88(6):062709.
12. Bramer D, Wei GW. Atom-specific persistent homology and its application to protein flexibility analysis. Comput Math Biophys. 2020;8(1):1-35.
13. Gräßler J, Koschützki D, Schreiber F. CentiLib: Comprehensive analysis and exploration of network centralities. Bioinformatics. 2012;28(8):1178-1179.
14. Hu Z, Hung JH, Wang Y, Chang YC, Huang CL, Huyck M, et al. VisANT 3.5: Multi-scale network visualization, analysis and inference based on the gene ontology. Nucleic Acids Res. 2009;37(suppl_2):W115-W121.
15. Lee MS, Ji QC. Protein analysis using mass spectrometry: Accelerating protein biotherapeutics from lab to patient. John Wiley & Sons. 2017.
16. Opron K, Xia K, Burton Z, Wei GW. Flexibility–rigidity index for protein–nucleic acid flexibility and fluctuation analysis. J Comput Chem. 2016;37(14):1283-1295.
17. Kovacev-Nikolic V, Bubenik P, Nikolic D, Heo G. Using persistent homology and dynamical distances to analyze protein binding. Stat Appl Genet Mol Biol. 2016;15(1):19-38.